VR4101™

64-BIT MICROPROCESSOR

(PRELIMINARY)

μPD30101

VR4101 (PRELIMINARY)

VR4101UM(PRELIMINARY)
(U12149EJ1V0UM00, February 1997 NS)

PREFACE

p.1 p.548
FACSIMILE

VR4000, VR4100, VR4101, VR4200, VR4400, and VR-Series are trademarks of NEC Corporation.
R4000 is a trademark of MIPS Computer Systems, Inc.
MIPS is a trademark of MIPS Technologies, Inc.
iAPX is a trademark of Intel Corp.
DEC VAX is a trademark of Digital Equipment Corp.
UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

The information in this document is subject to change without notice.
No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.
NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.
While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.
NEC devices are classified into the following three quality grades:
"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots

Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)

Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's Data Sheets or Data Books.
If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.
Anti-radioactive design is not implemented in this product.

M7 96.5

NOTES FOR CMOS DEVICES

PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to VDD or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

[MEMO]

PREFACE

Readers

This manual targets users who intends to understand the functions of the VR4101 and to design application systems using this microprocessor.

Purpose

This manual introduces the architecture and hardware functions of the VR4101 to users, following the organization described below.

Organization

This manual consists of the following contents:

- Introduction
- Pipeline operation
- Cache organization and memory management system
- Exception processing
- Initialization interface
- Interrupts
- Peripheral units
- Instruction set details

How to read this manual

It is assumed that the reader of this manual has general knowledge in the fields of electric engineering, logic circuits, and microcomputers.

The VR4000™ in this manual includes the VR4400™.

To learn about detailed function of a specific instruction,
-> Read **Chapter 2 CPU Instruction Set Summary** and **Chapter 24 CPU Instruction Set Details**.

To learn about the overall functions of the VR4101,
-> Read this manual in sequential order.

To learn about electrical specifications,
-> Refer to **Data Sheet** which is separately available.

Legend

Data significance:	Higher on left and lower on right
Active low:	XXX* (trailing asterisk after pin and signal names)
Numeric representation:	binary ... XXXX or $XXXX_2$
	decimal ... XXXX
	hexadecimal ... 0xXXXX

Prefixes representing an exponent of 2 (for address space or memory capacity):

K (kilo)	$2^{10} = 1024$
M (mega)	$2^{20} = 1024^2$
G (giga)	$2^{30} = 1024^3$
T (tera)	$2^{40} = 1024^4$
P (peta)	$2^{50} = 1024^5$
E (exa)	$2^{60} = 1024^6$

Related Documents

The related documents indicated here may include preliminary version. However, preliminary versions are not marked as such.

- User’s manual

VR4101 User’s Manual	This manual
VR4100™ User’s Manual	U10050E

- Data sheet

VR4101 Data Sheet	U11846E
VR4100 Data Sheet	U10428E

- Application note

VR4101 Application Note	To be issued

SUMMARY OF CONTENTS

[MEMO]

TABLE OF CONTENTS

[MEMO]

LIST OF FIGURES (1/8)

LIST OF FIGURES (2/8)

LIST OF FIGURES (3/8)

LIST OF FIGURES (4/8)

LIST OF FIGURES (5/8)

LIST OF FIGURES (6/8)

LIST OF FIGURES (7/8)

LIST OF FIGURES (8/8)

LIST OF TABLES (1/4)

LIST OF TABLES (2/4)

LIST OF TABLES (3/4)

LIST OF TABLES (4/4)

CHAPTER 1 INTRODUCTION

The VR4101 is one of the RISC (reduced instruction set computer) microprocessor VR-Series products manufactured by NEC. It is designed around the RISC architecture developed by MIPS. This 64-bit microprocessor mainly consists of the VR4100 CPU core (containing a cache memory, high-speed sum-of-products operation unit, and address management unit), DMA, and peripheral circuit interface units (such as a serial interface, keyboard interface, IrDA interface, touch panel interface, and real-time clock) required by battery-driven information units.

The VR4101 microprocessor is compatible with the MIPS I, MIPS II, and MIPS III Instruction Set Architecture (ISA). However, none of the floating-point, LL, LLD, SC and SCD instructions is supported.

This microprocessor does not provide on-chip support for a secondary cache or multiprocessing, and floating-point operation.

1.1 CHARACTERISTICS

The VR4100 has the following characteristics:

- MIPS I, II, III instruction sets (without the FPU, LL, and SC instructions from the VR4000 family instruction set) compatible
- Internal 64-bit processing
- 32-bit physical address space and 40-bit virtual address space
- Internal operating frequency: 33 MHz
- Optimized 5-stage pipeline, 2-Kbyte instruction cache and 1-Kbyte data cache, and 32-double-entry TLB
- Write-back cache for reducing store operations that use the system bus
- madd16 and dmadd16 instructions for executing a sum-of-products operation of 16-bit data x 16-bit data + 64-bit data within one clock cycle
- Effective power management features, which include the following four operating modes:
    - Full Speed mode
    - Standby mode
    - Suspend mode
    - Hibernate mode
- No floating-point functions
- No secondary cache, multiprocessor function (LL, LLD, SC, or SCD instruction)
- All clock pulses for internal operations generated from a 32-kHz crystal
- Built-in clock generator
- Built-in PLL for frequency multiplication by 2024
- External bus frequency of 16 MHz
- Built-in 8-Mbyte DRAM and 16-Mbyte masked ROM interfaces
- Built-in LCD, keyboard, and touch panel interfaces
- Built-in 5-channel DMA controller

- Built-in serial and debug serial interfaces
- Built-in IrDA controller
- ISA bus-subset supported
- 160-pin low-profile plastic QFP (LQFP)

1.2 ORDERING INFORMATION

Part Number	Package
µPD30101GM-33-8ED	160-pin plastic LQFP (fine pitch) (24 x 24 mm)

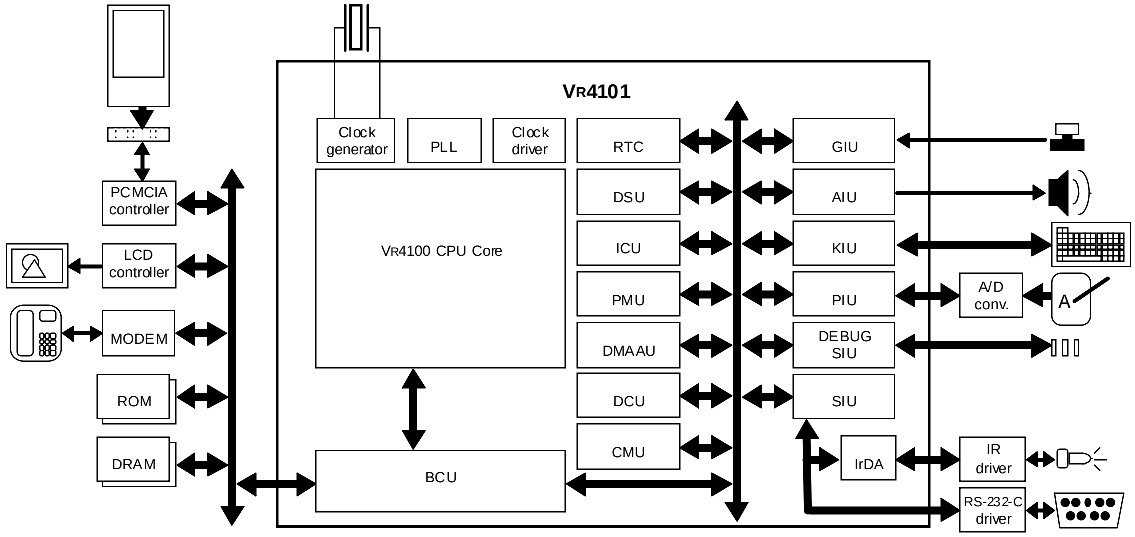
1.3 64-BIT ARCHITECTURE

The VR4101 microprocessor has a 64-bit architecture. However, it can run 32-bit applications.

1.4 VR4101 PROCESSOR

Figure 1-1 is an internal block diagram of the VR4101 processor. Figure 1-2 is a block diagram showing the internal structure of the VR4100 CPU core.

Figure 1-1. VR4101 Internal Block Diagram and Example of Connection to External Blocks



1.4.1 Internal Structure

This section introduces each unit in the VR4101.

(1) Bus control unit (BCU)

In the VR4101, the bus control unit (BCU) transfers data between the VR4100 CPU core and SysAD bus. It also controls external circuits, such as the LCD controller connected to the system bus, DRAM, ROM (flash memory or masked ROM), and PCMCIA controller, and transfers data between the VR4101 and these external devices, using the address and data buses.

(2) Real-time clock (RTC)

The real-time clock (RTC) is provided with an accurate counter that operates on a 32.768-kHz clock pulse supplied from the clock generator. It is also provided with several counters and Compare registers for controlling various interrupts.

(3) Deadman’s switch (DSU)

The Deadman’s switch unit (DSU) is used to check whether the processor is running normally. If the register of this unit is not cleared by software within a specified period, the system is shut down.

(4) Interrupt control unit (ICU)

The interrupt control unit (ICU) controls interrupts that are caused by factors either internal or external to the VR4101, and informs the VR4100 CPU core when an interrupt occurs.

(5) Power management unit (PMU)

The power management unit (PMU) outputs signals necessary to control the power of the entire system including the VR4101. The signals are used to control the PLL of the VR4100 CPU core and the internal clocks (PClock, TClock, and MasterOut) in low-power modes.

(6) Direct memory access address unit (DMAAU)

The direct memory access address unit (DMAAU) controls the address of five different DMA transfers.

(7) Direct memory access control unit (DCU)

The direct memory access control unit (DCU) controls the arbitration of five different DMA transfers.

(8) Clock mask unit (CMU)

The clock mask unit (CMU) controls the way the clocks TClock and MasterOut are supplied from the VR4100 CPU core to internal peripheral units.

(9) General purpose I/O unit (GIU)

Basically, the general purpose I/O unit (GIU) controls 12 GPIO pins. Among the 12 GPIO pins of the current VR4101 version, some pins are controlled directly by other units.

(10) Audio interface unit (AIU)

The audio interface unit (AIU) generates sound having a specified frequency, using a PWM, and outputs sound signals. Buzzer output is also available.

(11) Keyboard interface unit (KIU)

The keyboard interface unit (KIU) has 8 scan lines and as many detection lines. It can detect when any of 64 keys are pressed. It supports key rollover for two to three continuous strokes.

(12) Touch panel interface unit (PIU)

The touch panel interface unit (PIU) detects when the touch panel is touched. The current VR4101 version supports interfaces for two A/D converters, TLC2543C and TLV1543C.

(13) Debug serial interface unit (Debug SIU)

The debug serial interface unit (debug SIU) is a serial interface for debugging. It supports a maximum transfer rate of 115 kbps.

(14) Serial interface unit (SIU)

The serial interface unit (SIU) complies with the RS-232-C specification. It supports a maximum transfer rate of 115 kbps. Also available is an IrDA serial interface supporting a maximum transfer rate of 115 kbps, but this interface and the RS-232-C interface are mutually exclusive.

1.4.2 I/O registers

The I/O registers are used for peripheral unit control. The I/O registers are listed below.

Table 1-1. BCU Registers

Name	Function	Address
BCUCNTREG	BCU Control register	0x0B00 0000
BCUBRREG	BCU Bus Restrain register	0x0B00 0002
BCUBRCNTREG	BCU Bus Restrain Count register	0x0B00 0004
BCUBCLREG	BCU CPU Restrain Disable register	0x0B00 0006
BCUCLCNTREG	BCU CPU Restrain Disable Count register	0x0B00 0008
BCUSPEEDREG	BCU Access Cycle Change register	0x0B00 000A
BCUERRSTREG	BCU Bus Error Status register	0x0B00 000C
BCURFCNTREG	BCU Refresh Control register	0x0B00 000E
PREVIDREG	Peripheral Revision ID register	0x0B00 0010

Table 1-2. DMAAU Registers

Name	Function	Address
PADDMAADRLREG	PAD1 DMA Address register Low	0x0B00 0020
PADDMAADRHREG	PAD1 DMA Address register High	0x0B00 0022
SRXDMAADRLREG	SRX1 DMA Address register Low	0x0B00 0024
SRXDMAADRHREG	SRX1 DMA Address register High	0x0B00 0026
STXDMAADRLREG	STX1 DMA Address register Low	0x0B00 0028
STXDMAADRHREG	STX1 DMA Address register High	0x0B00 002A
AUDDMAADRLREG	AUDIO1 DMA Address register Low	0x0B00 002C
AUDDMAADRHREG	AUDIO1 DMA Address register High	0x0B00 002E
KEYDMAADRLREG	KEY1 DMA Address register Low	0x0B00 0030
KEYDMAADRHREG	KEY1 DMA Address register High	0x0B00 0032

Table 1-3. DCU Registers

Name	Function	Address
DMARSTREG	DMA Reset register	0x0B00 0040
DMAIDLEREG	DMA Idle register	0x0B00 0042
DMASENREG	DMA Sequencer Enable register	0x0B00 0044
DMAMSKREG	DMA Mask register	0x0B00 0046
DMAREQREG	DMA Request register	0x0B00 0048

Table 1-4. CMU Register

Name	Function	Address
CMUCLKMSKREG	CMU Clock Mask register	0x0B00 0060

Table 1-5. ICU Registers

Name	Function	Address
SYSINTREG	Level 1 System register	0x0B00 0080
PIUINTREG	Level 2 PIU register	0x0B00 0082
AUDINTREG	Level 2 AUD register	0x0B00 0084
KIUINTREG	Level 2 KIU register	0x0B00 0086
GIUINTREG	Level 2 GIU register	0x0B00 0088
SIUINTREG	Level 2 SIU register	0x0B00 008A
MSYSINTREG	Level 1 Mask System register	0x0B00 008C
MPIUINTREG	Level 2 Mask PIU register	0x0B00 008E
MADUINTREG	Level 2 Mask AUD register	0x0B00 0090
MKIUINTREG	Level 2 Mask KIU register	0x0B00 0092
MGIUINTREG	Level 2 Mask GIU register	0x0B00 0094
MSIUINTREG	Level 2 Mask SIU register	0x0B00 0096
NMIREG	NMI register	0x0B00 0098
SOFTINTREG	Software Interrupt register	0x0B00 009A

Table 1-6. PMU Registers

Name	Function	Address
PMUINTREG	PMU Interrupt/Status register	0x0B00 00A0
PMUCNTREG	PMU Control register	0x0B00 00A2

Table 1-7. RTC Registers

Name	Function	Address
ETIMELREG	Elapsed Time L register	0x0B00 00C4
ETIMEMREG	Elapsed Time M register	0x0B00 00C6
ETIMEHREG	Elapsed Time H register	0x0B00 00C8
ECMPHREG	Elapsed Compare H register	0x0B00 00CA
ECMPLREG	Elapsed Compare L register	0x0B00 00CC
ECMPMREG	Elapsed Compare M register	0x0B00 00CE
RTCLLREG	RTC Long L register	0x0B00 00D0
RTCLHREG	RTC Long H register	0x0B00 00D2
RTCLCNTLREG	RTC Long Count L register	0x0B00 00D4
RTCLCNTHREG	RTC Long Count H register	0x0B00 00D6
TCLKCNTLREG	TCLK Count L register	0x0B00 00D8
TCLKCNTHREG	TCLK Count H register	0x0B00 00DA
RTCINTREG	RTC Interrupt register	0x0B00 00DC

Table 1-8. DSU Registers

Name	Function	Address
DSUCNTREG	DSU Control register	0x0B00 00E0
DSUSETREG	DSU Dead Time Set register	0x0B00 00E2
DSUCLRREG	DSU Clear register	0x0B00 00E4
DSUTIMREG	DSU Elapsed Time register	0x0B00 00E6

Table 1-9. GIU Registers

Name	Function	Address
GOUTENREG	GPIO Output Enable register	0x0B00 0100
GPOTDATREG	GPIO Port Data register	0x0B00 0102
GINTSTREG	GPIO Interrupt Status register	0x0B00 0104
GINTENREG	GPIO Interrupt Enable register	0x0B00 0106
GCINTSREG	GPIO Change Point Interrupt register	0x0B00 0108
GLINTSREG	GPIO Interrupt Level Specified register	0x0B00 010A

Table 1-10. PIU Registers

Name	Function	Address
PIUDATAREG	PIU Touch Panel Point Data register	0x0B00 0120
PIUCNTREG	PIU Control register	0x0B00 0122
PIUINTREG	PIU Interrupt Cause register	0x0B00 0124
PIUSIVLREG	PIU Data Sampling Interval register	0x0B00 0126
PIUSTBLREG	PIU AD Converter Start Delay register	0x0B00 0128
PIUCMDREG	PIU AD Command register	0x0B00 012A
PIUCIVLREG	PIU AD Check Interval register	0x0B00 013C

Table 1-11. SIU Registers

Name	Function	Address
SIURXDATREG	SIU Rx Data register	0x0B00 0140
SIUTXDATREG	SIU Tx Data register	0x0B00 0142
SIUCNTREG	SIU Control register	0x0B00 0144
SIUDLENGTHREG	SIU RxTx Data Length register	0x0B00 0146
SIUINTREG	SIU Interrupt register	0x0B00 0148
SIURS232CREG	SIU RS-232-C Control register	0x0B00 014A
SIUBAUDSELREG	SIU Baud rate Select register	0x0B00 014C

Table 1-12. AIU Registers

Name	Function	Address
AIUDATREG	AIU Data register	0x0B00 0162
AIURESETREG	AIU Reset	0x0B00 0164
AIUMODEREG	AIU Mode Select	0x0B00 0166
AIUSEQENREG	AIU Sequencer Enable	0x0B00 0168
AIUMUTEREG	AIU Mute Control	0x0B00 016A
AIUSTATREG	AIU Status	0x0B00 016C
AIUSTPPAGEREG	AIU DMA Stop at Page	0x0B00 016E
AIUVALIDREG	AIU Counter Valid Bits	0x0B00 0170
AIUINTREG	AIU Interrupts	0x0B00 0172
AIUCOUNT0REG	AIU Counter 0	0x0B00 0174
AIUCOUNT1REG	AIU Counter 1	0x0B00 0176
AIUREPNUMREG	AIU PWM Repeat Number	0x0B00 0178
AIUBUSENREG	AIU Bus IF Enable	0x0B00 017A

Table 1-13. KIU Registers

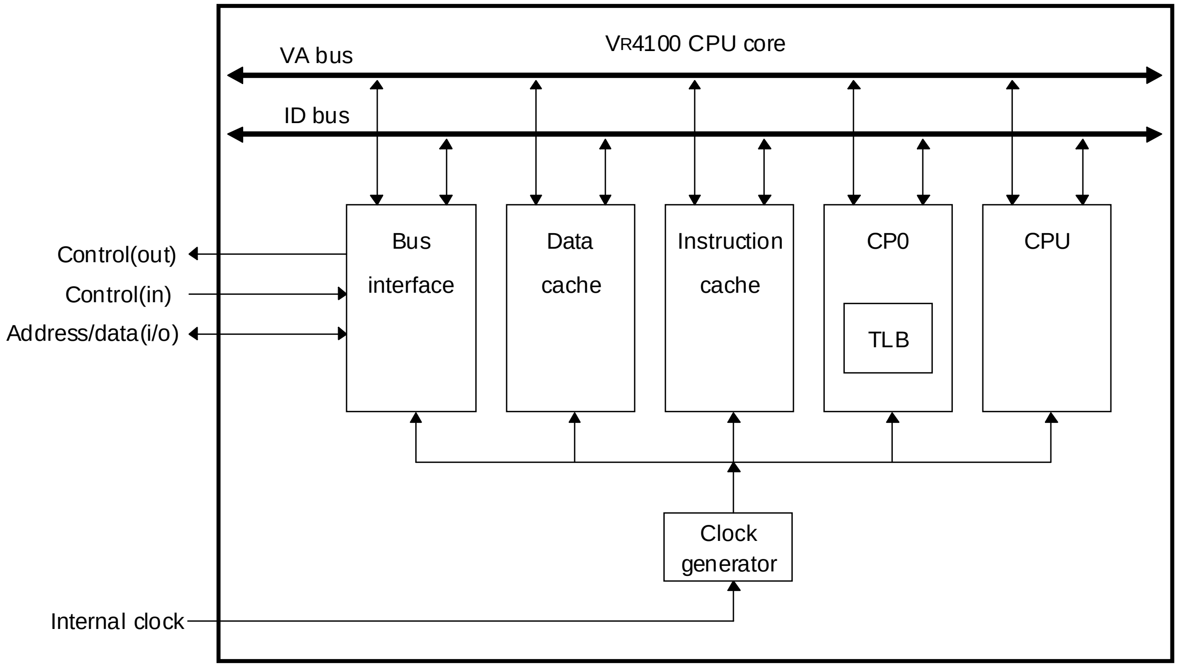
Name	Function	Address
KIUDATREG	KIU Key Data register	0x0B00 0180
KIUASCANREG	KIU Key Auto Scan register	0x0B00 0184
KIUASTOPREG	KIU Key Auto Stop register	0x0B00 0186
KIUSCANREG	KIU Key Scan register	0x0B00 0188
KIUSTOPREG	KIU Key Stop register	0x0B00 018A
KIUSAPREG	KIU Key Stop at Page register	0x0B00 018C
KIUSCANSREG	KIU Scan Status register	0x0B00 018E
KIUWKSREG	KIU Wait Key Scan Stable register	0x0B00 0190
KIUWKIREG	KIU Wait Key Scan Interval register	0x0B00 0192
KIUSRNREG	KIU Stop Repeat Number register	0x0B00 0194
KIUINTREG	KIU Interrupt register	0x0B00 0196
KIURSTREG	KIU Reset register	0x0B00 0198
KIUENREG	KIU Enable register	0x0B00 019A
DOZEKEYINTREG	DOZE Key Interrupt register	0x0B00 019C
EVVOLREG	EVVOL register	0x0B00 019E

Table 1-14. DebugSIU Registers

Name	Function	Address
ASIM00REG	Asynchronous Mode 0 register	0x0B00 01A4
ASIM01REG	Asynchronous Mode 1 register	0x0B00 01A6
RXB0RREG	Receive Buffer register (Extended)	0x0B00 01A8
RXB0LREG	Receive Buffer register	0x0B00 01AA
TXS0RREG	Transmit Data register (Extended)	0x0B00 01AC
TXS0LREG	Transmit Data register	0x0B00 01AE
ASIS0REG	Status register	0x0B00 01B0
INTR0REG	Debug SIU Interrupt register	0x0B00 01B2
BPRM0REG	Baud rate Generator Prescaler Mode register	0x0B00 01B6
DSIURESETREG	Debug SIU Reset register	0x0B00 01B8

1.5 VR4100 CPU CORE

Figure 1-2. VR4100 CPU Core Internal Block Diagram



1.5.1 Internal Structure

- CPU

  CPU has the hardware resources to execute integer instructions. It has a 64-bit register file, 64-bit integer data path, and sum-of-products operation unit.
- Coprocessor 0 (CP0)

  Coprocessor 0 (CP0) has the memory management unit (MMU) and handles exception processing. The MMU handles address translation and checks memory accesses that occur between different memory segments (user, supervisor, or kernel). The translation lookaside buffer (TLB) is used to translate virtual to physical addresses.
- Instruction cache

  Instruction cache is direct-mapped, virtually-indexed, and physically-tagged. Its capacity is 2K bytes.
- Data cache

  Data cache is a direct-mapped, virtually-indexed, and physically-tagged write-back cache. Its capacity is 1K bytes.
- CPU bus interface

  The CPU bus interface controls data transfer between the VR4100 CPU core and the BCU peripheral unit. The VR4100 CPU core bus interface consists of 32-bit input and output multiplexed address/data buses used for transferring clock and interrupt control signals.
- Clock generator

  The output frequency of the 32.768-kHz crystal is received at the internal oscillation circuit, where it is multiplied by 1012 using a phase-lock loop (PLL) to generate the pipeline clock (PClock) pulse. The PClock pulse is in turn used to generate the internal bus clocks (TClock and MasterOut) pulse.

1.5.2 CPU Registers

The VR4100 CPU core provides registers as below.

- 32 x 64-bit general-purpose registers (GPRs)

In addition, the processor provides the following special registers:

- 64-bit Program Counter (PC)
- 64-bit HI register, containing the integer multiply and divide upper doubleword result
- 64-bit LO register, containing the integer multiply and divide lower doubleword result

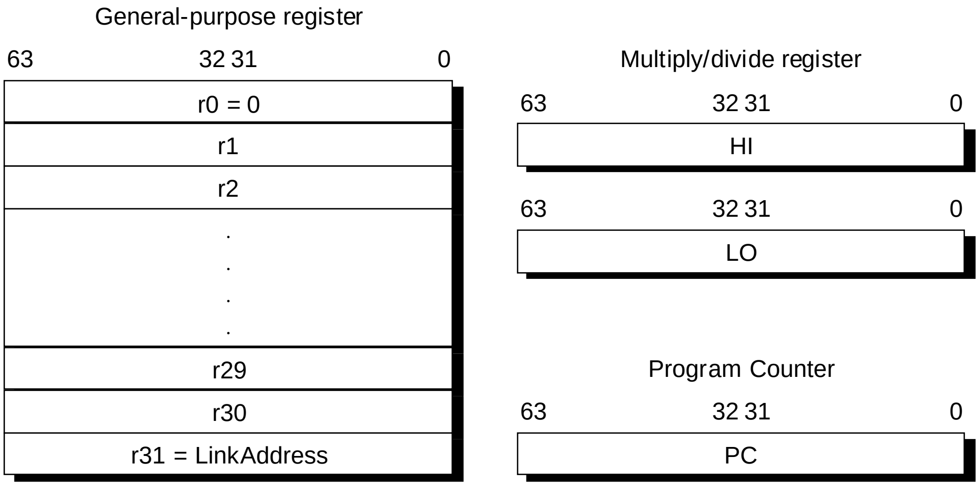
Two of the general-purpose registers have assigned functions:

- r0 is hardwired to a value of zero, and can be used as the target register for any instruction whose result is to be discarded. r0 can also be used as a source when a zero value is needed.
- r31 is the link register used by Jump and Link (JAL/JALR) instructions. This register can be used for other instructions. However, be careful that use of the register by a link instruction will not coincide with use of the register for other operations.

CPU registers can operate as either 32-bit or 64-bit registers, depending on the VR4101 processor mode of operation.

Figure 1-3 shows the VR4101 CPU registers.

Figure 1-3. VR4101 CPU Registers



The VR4101 has no Program Status Word (PSW) register as such; this is covered by the Status and Cause registers incorporated within the System Control Coprocessor.

The CP0 registers are used for exception processing and address management. The CP0 registers are briefly explained later in this chapter.

1.5.3 CPU Instruction Set Overview

Each CPU instruction is 32 bits long. As shown in Figure 1-4, there are three instruction formats:

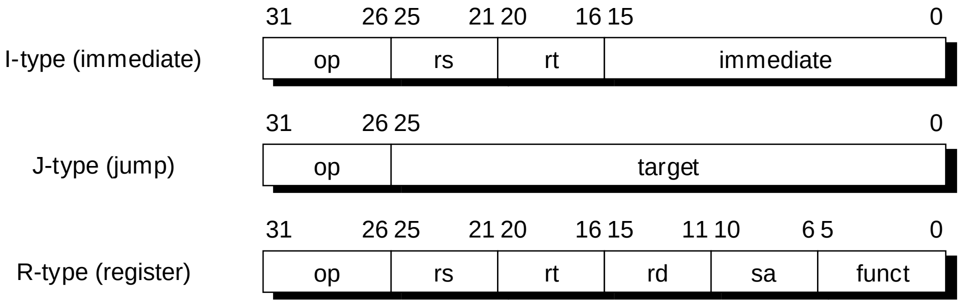
✧ immediate (I-type)

✧ jump (J-type)

✧ register (R-type)

The instruction set is divided into several groups as shown below. Fields of the instruction formats are described in Chapter 2.

Figure 1-4. CPU Instruction Formats



Instruction decoding is greatly simplified by limiting the number of formats to these three. This limitation means that the more complicated (and less frequently used) operations and addressing modes can be synthesized by the compiler, using sequences of these same simple instructions.

The instruction set can be further divided into the following groupings:

- Load and store instructions move data between memory and general-purpose registers. They are all immediate (I-type) instructions, since the only addressing mode supported is base register plus 16-bit, signed immediate offset.
- Computational instructions perform arithmetic, logical, shift, multiply, and divide operations on values in registers. They include R-type (in which both the operands and the result are stored in registers) and I-type (in which one operand is a 16-bit signed immediate value) formats.
- Jump and branch instructions change the control flow of a program. Jumps are always made to an absolute address formed by combining a 26-bit target address with the high-order bits of the Program Counter (J-type format) or register address (R-type format). The format of the branch instructions is I type. Branches have 16-bit offsets relative to the Program Counter. JAL instructions save their return address in register 31.
- Coprocessor 0 (System Control Coprocessor, CP0) instructions perform operations on CP0 registers to control the memory-management and exception-handling facilities of the processor.
- Special instructions perform system calls and breakpoint operations, or cause a branch to the general exception-handling vector based upon the result of a comparison. These instructions occur in both R-type (both the operands and the result are stored in registers) and I-type (one operand is a 16-bit signed immediate value) formats.

Chapter 2 provides a more detailed summary (Refer to the Chapter 24 for detailed descriptions of the operation of each instruction) .

1.5.4 Data Formats and Addressing

The VR4101 uses following four data formats:

- Doubleword (64 bits)
- Word (32 bits)

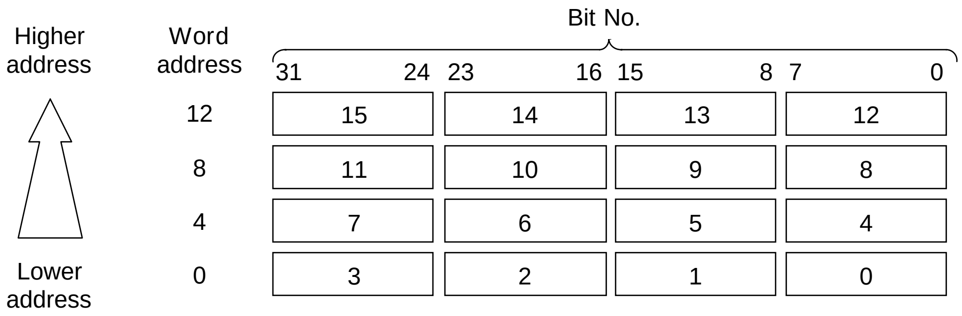
- Halfword (16 bits)
- Byte (8 bits)

For the VR4100 CPU core, byte ordering within all of the larger data formats - halfword, word, doubleword - can be configured in either big-endian or little-endian order. **However, the VR4101 supports the little-endian order only.**

Endianness refers to the location of byte 0 within the multi-byte data structure. Figure 1-5 shows the ordering of bytes within words and the ordering of words within doubleword structures for the little-endian conventions.

When configured as a little-endian system, byte 0 is always the least-significant (rightmost) byte, which is compatible with iAPX™ and DEC VAX™ conventions. Figure 1-5 shows this configuration.

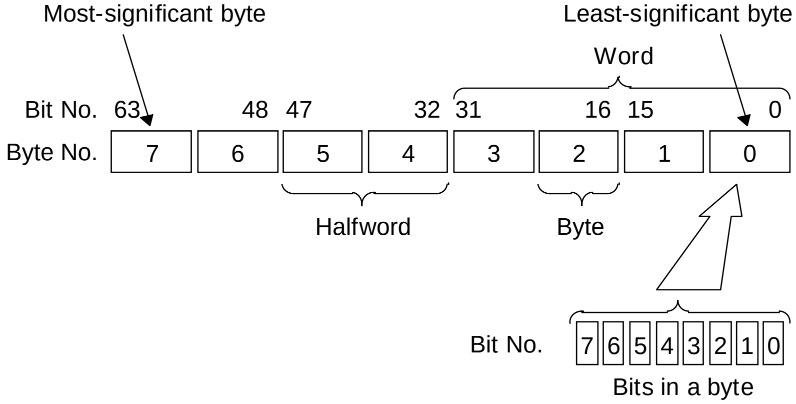
Figure 1-5. Little-Endian Byte Ordering



In this manual, bit 0 is always the least-significant (rightmost) bit; thus, bit designations are always little-endian.

Figure 1-6 shows little-endian byte ordering in doublewords.

Figure 1-6. Little-Endian Data in a Doubleword



The CPU uses following byte boundaries for halfword, word, and doubleword accesses:

- Halfword: An even byte boundary (0, 2, 4...)
- Word: A byte boundary divisible by four (0, 4, 8...)
- Doubleword: A byte boundary divisible by eight (0, 8, 16...)

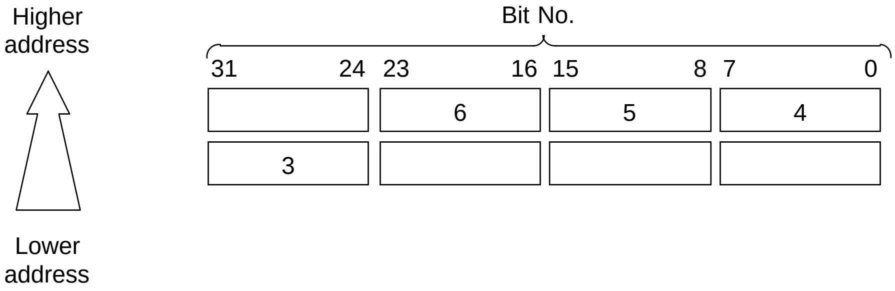
The following special instructions to load and store data that are not aligned on 4-byte (word) or 8-byte (doubleword) boundaries:

LWL	LWR	SWL	SWR
LDL	LDR	SDL	SDR

These instructions are used in pairs to provide an access to misaligned data. Accessing misaligned data incurs one additional instruction cycle over that required for accessing aligned data.

Figure 1-7 shows the access of a misaligned word that has byte address 3 for the little-endian conventions.

Figure 1-7. Misaligned Word Accessing (Little-Endian)



1.5.5 Coprocessors (CP0-CP3)

The MIPS ISA defines four coprocessors (designated CP0, CP1, CP2, and CP3):

- CP0 is incorporated on the CPU chip and supports the virtual memory system and exception handling. The virtual memory system is implemented using an on-chip TLB and the CP0 registers. CP0 is also referred to as the System Control Coprocessor.
- CP1 is reserved for floating-point instructions.
- CP2 is reserved for future definition by MIPS.
- CP3 is no longer defined. CP3 instructions are reserved for future extensions.

CP0 is described in Chapter 4 and Chapter 5.

(1) System Control Coprocessor (CP0)

CP0 translates virtual addresses into physical addresses and manages exceptions and transitions between kernel, supervisor, and user states.

CP0 also controls the cache system, as well as providing diagnostic control and error recovery facilities.

The CP0 registers shown in Figure 1-8 and described in Table 1-15 manipulate the memory-management and exception-handling capabilities of the CP0.

Figure 1-8. CP0 Registers

Register No.	Register name	Register No.	Register name
0	Index*	16	Config*
1	Random*	17	LLAddr*
2	EntryLo0*	18	WatchLo**
3	EntryLo1*	19	WatchHi**
4	Context**	20	XContext**
5	PageMask*	21	-
6	Wired*	22	-
7	-	23	-
8	BadVAddr**	24	-
9	Count**	25	-
10	EntryHi*	26	PErr**
11	Compare**	27	CacheErr**
12	Status**	28	TagLo*
13	Cause**	29	TagHi*
14	EPC**	30	ErrorEPC**
15	PRId*	31	-

*	for Memory management
**	for Exception handling
-	Reserved

Table 1-15. System Control Coprocessor (CP0) Register Definitions

Number	Register	Description
0	Index	Programmable pointer to TLB array
1	Random	Pseudo-random pointer to TLB array (read only)
2	EntryLo0	Low half of TLB entry for even VPN
3	EntryLo1	Low half of TLB entry for odd VPN
4	Context	Pointer to kernel virtual PTE in 32-bit mode
5	PageMask	TLB page mask
6	Wired	Number of wired TLB entries
7	—	Reserved
8	BadVAddr	Virtual address where the most recent error occurred
9	Count	Timer count
10	EntryHi	High half of TLB entry
11	Compare	Timer compare
12	Status	Status register
13	Cause	Cause of last exception
14	EPC	Exception Program Counter
15	PRId	Processor revision identifier
16	Config	Configuration register
17	LLAddr	Reserved
18	WatchLo	Memory reference trap address low bits
19	WatchHi	Memory reference trap address high bits
20	XContext	Pointer to kernel virtual PTE in 64-bit mode
21-25	—	Reserved
26	PErr	Cache parity bits
27	CacheErr	Index and status of cache error
28	TagLo	Cache Tag register (low)
29	TagHi	Cache Tag register (high)
30	ErrorEPC	Error Exception Program Counter
31	—	Reserved

1.5.6 Floating-Point Unit (FPU)

The VR4101 does not support the floating-point unit (FPU). Coprocessor Unusable exception will occur if any FPU instructions are executed. If necessary, FPU instructions should be emulated by software in an exception handler.

1.5.7 Cache

The VR4101 chip incorporates instruction and data caches, which are independent of each other. This configuration enables high-performance pipeline operations. Both caches have a 64-bit data bus. These buses can be accessed in parallel. The instruction cache of the VR4101 has a storage capacity of 2 KB, while the data cache has a capacity of 1 KB.

(1) Instruction cache

The VR4101 incorporates a direct-mapped on-chip instruction cache. This virtually indexed, physically tagged cache is 2 KB in size and is protected with word parity.

Because the cache is virtually indexed, the virtual-to-physical address translation occurs in parallel with the cache access. The tag holds a 22-bit physical address and valid bit, and is parity protected.

The instruction cache is 64-bits wide, and can be refilled or accessed in a single pipeline cycle. Instruction fetches require only 32 bits per cycle, for a maximum transfer rate of 132 MB/sec. The line size is four words (16 bytes).

(2) Data cache

For single cycle data access, the VR4101 includes a 1 KB on-chip data cache that is directly-mapped with a fixed 16-byte (four words) line size.

The data cache is protected with byte parity and its tag is protected with a single parity bit. It is virtually indexed and physically tagged to allow address translation and data cache access simultaneously.

The write policy is writeback, which means that storing data to a cache does not immediately cause main memory to be updated. This increases system performance by reducing bus traffic.

1.6 MEMORY MANAGEMENT SYSTEM (MMU)

The VR4101 has a 32-bit physical addressing range of 4 Gbytes. However, since it is rare for systems to implement a physical memory space as large as that memory space, the CPU provides a logical expansion of memory space by translating addresses composed in the large virtual address space into available physical memory addresses. The VR4101 supports the following two addressing modes:

- ✧ 32-bit mode, in which the virtual address space is divided into 2 Gbytes per user process and 2 Gbytes for the kernel.
- ✧ 64-bit mode, in which the virtual address is expanded to1 Tbyte (2^{40} bytes) of user virtual address space.

A detailed description of these address spaces is given in Chapter 4.

1.6.1 Translation Lookaside Buffer (TLB)

Virtual memory mapping is performed using the translation lookaside buffer (TLB). The TLB converts virtual addresses to physical addresses. It is provided on-chip. It runs by a full-associative method. It has 32 entries, each mapping a pair of pages having a variable size (1 KB to 256 KB).

(1) Joint TLB

For fast virtual-to-physical address decoding, the VR4101 uses a large, fully associative TLB which translates 64 virtual pages to their corresponding physical addresses. The TLB is organized as 32 pairs of even-odd entries, and maps a virtual address and address space identifier (ASID) into the 4-Gbyte physical address space.

The page size can be configured, on a per-entry basis, to map a page size of 1 KB to 256 KB. A CP0 register is loaded with the size of the page to be mapped, and that size is entered into the TLB when a new entry is written. Thus, operating systems can provide special purpose maps; for example, a typical frame buffer can be memory-mapped using only one TLB entry.

Translating a virtual address to a physical address begins by comparing the virtual address from the processor with the virtual addresses in the TLB; there is a match when the virtual page number (VPN) of the address is the same as the VPN field of the entry, and either the Global (G) bit of the TLB entry is set, or the ASID field of the virtual address is the same as the ASID field of the TLB entry.

This match is referred to as a TLB hit. If there is no match, a TLB Miss exception is taken by the processor and software is allowed to refill the TLB from a page table of virtual/physical addresses in memory.

1.6.2 Operating Modes

The VR4101 has three operating modes:

- ✧ User mode
- ✧ Supervisor mode
- ✧ Kernel mode

The manner in which memory addresses are translated or mapped depends on the operating modes; this is described in Chapter 4.

1.7 INSTRUCTION PIPELINE

The VR4101 has a 5-stage instruction pipeline. Under normal circumstances, one instruction is issued each cycle.

The instruction pipeline of the VR4101 operates at 33 MHz. The VR4101 achieves high throughput by shortening register access times and implementing virtually-indexed caches.

A detailed description of pipeline is provided in Chapter 3.

1.8 CLOCK INTERFACE

The VR4101 is provided with the following seven clocks.

- ✧ CLKX1, CLKX2 (input)

Clock inputs. Connect an oscillator having a frequency of 32.768 kHz to the CLKX1 and CLKX2 pins, or connect an external clock to the CLKX1 pin.

✧ RTC (internal)

Clock having a frequency of 32.768 kHz. This clock is generated from the clock input to the CLKX1 and CLKX2 pins. It is used in the PMU and the RTC units. In Hibernate mode, the internal clock of the CPU core is stopped and the VR4101 operates based on the RTC.

✧ PClock (internal)

Clock for the CPU core operation. This clock is generated from the clock input to the CLKX1 and CLKX2 pins multiplied at the PLL. It has a frequency of 33 MHz.

✧ MasterOut (internal)

Clock for the CPU core bus operation, and used for interrupt control. This clock has a frequency of 1/4 of the PClock frequency.

✧ TClock (internal)

Clock for the CPU core bus operation, the VR4101 internal bus operation, and operation of the peripheral units. This clock has a frequency of 1/2 of the PClock frequency.

✧ PCMCLK (output)

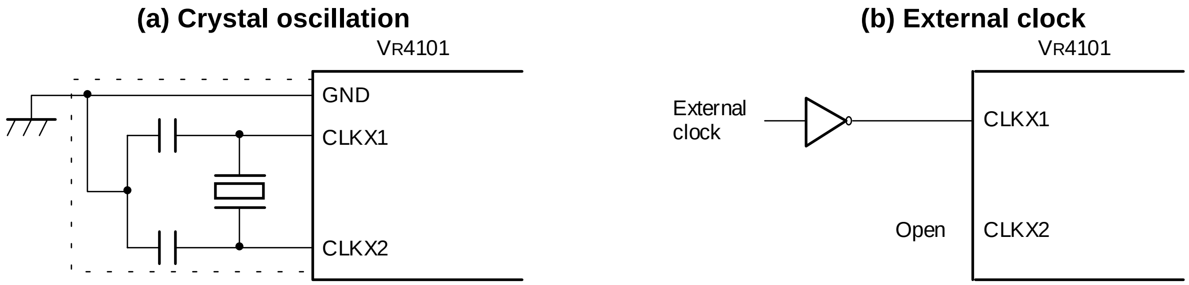
Clock supplied to the PCMCIA controller. This clock has a frequency of 8.25 MHz.

✧ ADCLK (output)

Clock supplied to the A/D converter. The frequency of this clock is set on the PIUSTBLREG register.

Figure 1-9 shows an external circuit of the clock oscillator.

Figure 1-9. External Circuit of Clock Oscillator



(a) Crystal oscillation

(b) External clock

Cautions 1. When using a clock oscillator, run wires in the area of this figure shown by broken lines, according to the following rules, to avoid effects such as stray capacitance:

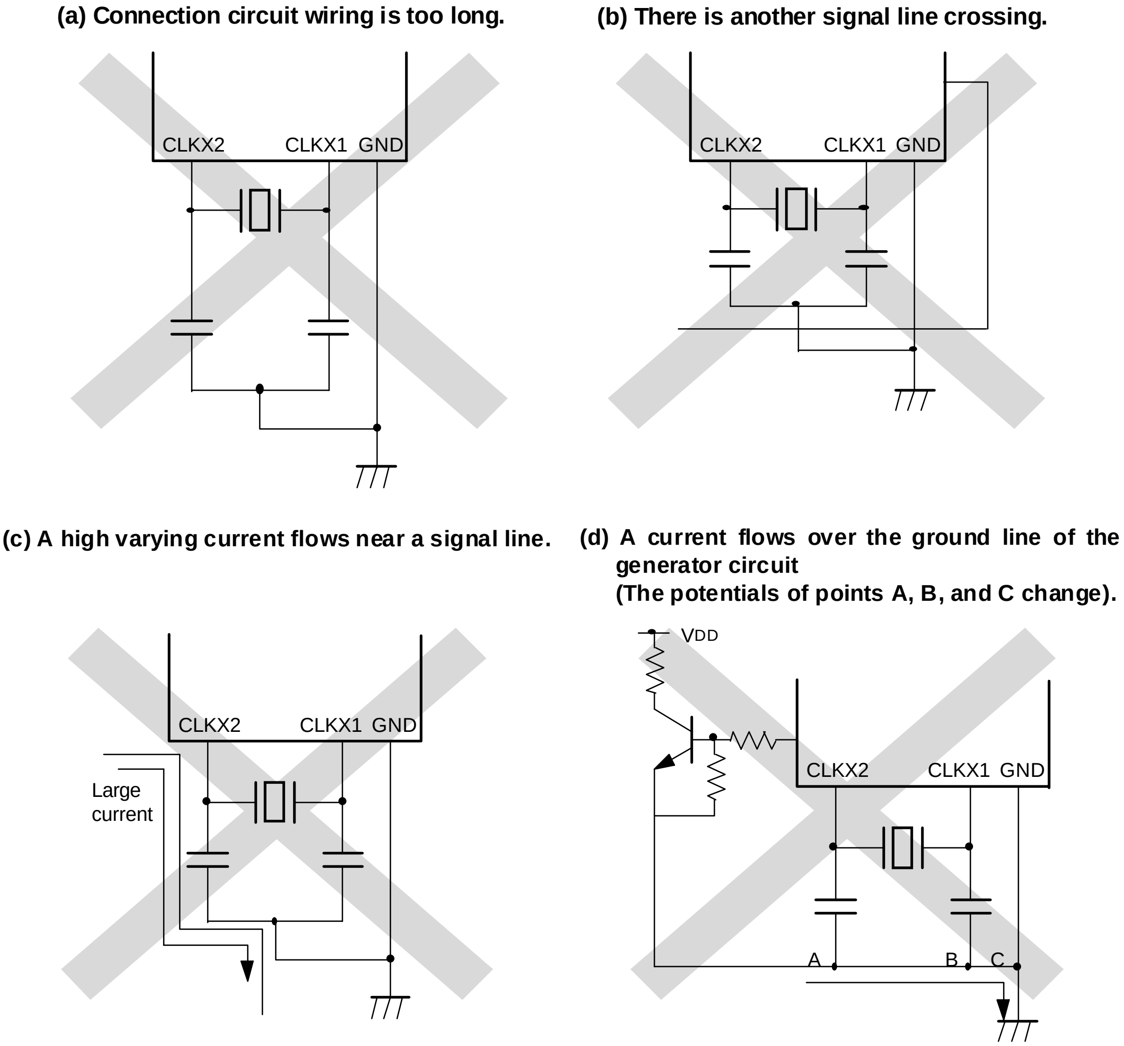
- **Minimize the wire.**
- **Never cause the wires to cross other signal lines or run near a line carrying a large varying current.**

- **Cause the grounding point of the capacitor of the oscillator circuit to have the same potential as GND. Never connect the capacitor to a ground pattern carrying a large current.**
- **Never extract a signal from the oscillator.**

2. **Take it into consideration that no capacitive load among wiring is applied to the CLKX2 pin when inputting an external clock.**

Figure 1-10 shows examples of oscillator having bad connection.

Figure 1-10. Examples of Oscillator with Bad Connection



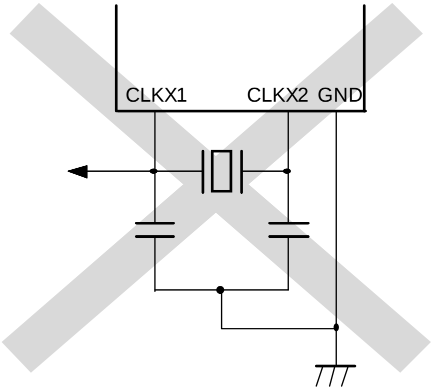
(a) Connection circuit wiring is too long.

(b) There is another signal line crossing.

(c) A high varying current flows near a signal line.

(d) A current flows over the ground line of the generator circuit (The potentials of points A, B, and C change).

(e) A signal is extracted.



[MEMO]

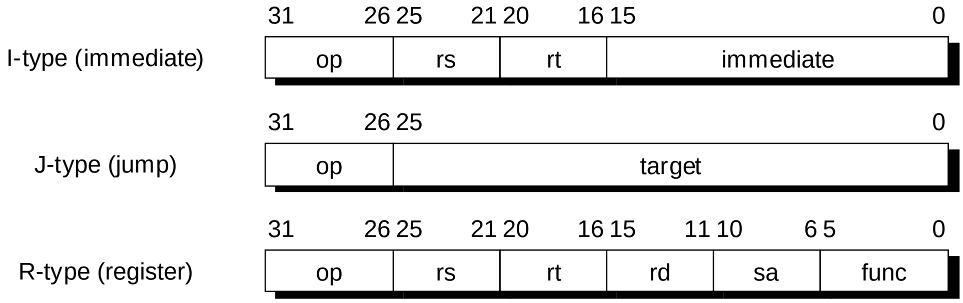
CHAPTER 2 CPU INSTRUCTION SET SUMMARY

This chapter is an overview of the central processing unit (CPU) instruction set; refer to the Chapter 24 for detailed descriptions of individual CPU instructions.

2.1 CPU INSTRUCTION FORMATS

Each CPU instruction consists of a single 32-bit word, aligned on a word boundary. There are three instruction formats - immediate (I-type), jump (J-type), and register (R-type) - as shown in Figure 2-1. The use of a small number of instruction formats simplifies instruction decoding, allowing the compiler to synthesize more complicated and less frequently used instruction and addressing modes from these three formats as needed.

Figure 2-1. CPU Instruction Formats



op:	6-bit operation code
rs:	5-bit source register specifier
rt:	5-bit target (source/destination) register or branch condition
immediate:	16-bit immediate value, branch offset or address offset
target:	26-bit jump target address
rd:	5-bit destination register specifier
sa:	5-bit shift amount
func:	6-bit function field

In the MIPS architecture, coprocessor instructions are implementation-dependent; refer to the Chapter 24 for details of individual coprocessor 0 instructions.

2.1.1 Support of the MIPS ISA

The VR4101 does not support a multiprocessor operating environment. Thus the synchronization support instructions defined in the MIPS II and MIPS III ISA - the load linked and store conditional instructions - cause reserved instruction exception. The LL bit is eliminated.

Note that the SYNC instruction is handled as a NOP instruction since all load/store instructions in this processor are executed in program order.

2.2 INSTRUCTION CLASSES

2.2.1 Load and Store Instructions

Load and store are immediate (I-type) instructions that move data between memory and the general-purpose registers. The only addressing mode that load and store instructions directly support is base register plus 16-bit signed immediate offset.

(1) Scheduling a load delay slot

A load instruction that does not allow its result to be used by the instruction immediately following is called a delayed load instruction. The instruction slot immediately following this delayed load instruction is referred to as the load delay slot.

In the VR4101, a load instruction can be followed directly by an instruction that accesses a register that is loaded by the load instruction. In this case, however, an interlock occurs for a necessary number of cycles. Any instruction can follow a load instruction, but the load delay slot should be scheduled appropriately for both performance and compatibility with other VR-Series microprocessors.

(2) Store delay slot

When a store instruction is writing data to a cache, the data cache is kept busy at the DC and WB stages. If an instruction (such as load) that follows directly the store instruction accesses the data cache in the DC stage, a hardware-driven interlock occurs. To overcome this problem, the store delay slot should be scheduled.

Table 2-1. Number of Delay Slot Cycles Necessary for Load and Store Instructions

Instruction	Necessary number of PCycles
Load	1
Store	1

(3) Defining access types

Access type indicates the size of a VR4101 processor data item to be loaded or stored, set by the load or store instruction opcode. Access types are defined in the Chapter 24.

Regardless of access type or byte ordering (endianness), the address given specifies the low-order byte in the addressed field. For a little-endian configuration, the low-order byte is the least-significant byte.

The access type, together with the three low-order bits of the address, define the bytes accessed within the addressed doubleword (shown in Table 2-2). Only the combinations shown in Table 2-2 are permissible; other combinations cause address error exceptions.

Refer to the Chapter 24 for individual descriptions of CPU load and store instructions.

Table 2-2. Byte Specification Related to Load and Store Instructions

Access type (value)	Low-order address bit			Accessed byte							
				Little endian							
	2	1	0	63							0
Doubleword (7)	0	0	0	7	6	5	4	3	2	1	0
7-byte (6)	0	0	0		6	5	4	3	2	1	0
	0	0	1	7	6	5	4	3	2	1	
6-byte (5)	0	0	0			5	4	3	2	1	0
	0	1	0	7	6	5	4	3	2		
5-byte (4)	0	0	0				4	3	2	1	0
	0	1	1	7	6	5	4	3			
Word (3)	0	0	0					3	2	1	0
	1	0	0	7	6	5	4				
Triple byte (2)	0	0	0						2	1	0
	0	0	1					3	2	1	
	1	0	0		6	5	4				
	1	0	1	7	6	5					
Halfword (1)	0	0	0							1	0
	0	1	0					3	2		
	1	0	0			5	4				
	1	1	0	7	6						
Byte (0)	0	0	0								0
	0	0	1							1	
	0	1	0						2		
	0	1	1					3			
	1	0	0				4				
	1	0	1			5					
	1	1	0		6						
	1	1	1	7							

2.2.2 Computational Instructions

Computational instructions can be either in register (R-type) format, in which both operands are registers, or in immediate (I-type) format, in which one operand is a 16-bit immediate.

Computational instructions perform the following operations on register values:

- ✧ Arithmetic
- ✧ Logical
- ✧ Shift
- ✧ Multiply
- ✧ Divide

These operations fit in the following four categories of computational instructions:

- ✧ ALU immediate instructions
- ✧ Three-operand register-type instructions
- ✧ Shift instructions
- ✧ Multiply and divide instructions

(1) 64-bit instructions

To maintain data compatibility between 64- and 32-bit modes, it is necessary to sign-extend 32-bit operands correctly. If the sign extension is not correct, the 32-bit operation result is meaningless.

(2) Cycle timing for multiply and divide instructions

MFHI and MFLO instructions (described in Chapter 24) after a multiply or divide instruction generate interlocks to delay execution of the next instruction, inhibiting the result from being read until the multiply or divide instruction completes.

Table 2-3 gives the number of processor cycles (PCycles) required to resolve interlock or stall between various multiply or divide instructions and a subsequent MFHI or MFLO instruction.

Table 2-3. Number of Stall Cycles in Multiply and Divide Instructions

Instruction	Number of instruction cycles
MULT	1
MULTU	1
DIV	35
DIVU	35
DMULT	4
DMULTU	4
DDIV	67
DDIVU	67
MADD16	1
DMADD16	1

For more information about computational instructions, refer to the individual instruction as described in Chapter 24.

2.2.3 Jump and Branch Instructions

Jump and branch instructions change the control flow of a program. All jump and branch instructions occur with a delay of one instruction: that is, the instruction immediately following the jump or branch instruction (this is known as the instruction in the delay slot) always executes while the target instruction is being fetched from memory.

For instructions involving a link (such as JAL and BLTZAL), the return address is saved in register r31.

Table 2-4. Number of Delay Slot Cycles in Jump and Branch Instructions

Instruction	Necessary number of cycles
Branch instruction	1
Jump instruction	1

(1) Overview of jump instructions

Subroutine calls in high-level languages are usually implemented with J or JAL instructions, both of which are J-type instructions. In J-type format, the 26-bit target address shifts left 2 bits and combines with the high-order 4 bits of the current program counter to form a 32-bit or 64-bit absolute address.

Returns, dispatches, and cross-page jumps are usually implemented with the JR or JALR instructions. Both are R-type instructions that take the 32-bit or 64-bit byte address contained in one of the general-purpose registers.

For more information, refer to Chapter 24.

(2) Overview of branch instructions

All branch instruction target addresses are computed by adding the address of the instruction in the delay slot to the 16-bit offset (shifted left 2 bits and sign-extended to 64 bits). All branches occur with a delay of one instruction.

If a branch likely instruction is not taken, the instruction in its delay slot is nullified. For all other branch instructions, the instruction in its delay slot is unconditionally executed.

For more information, refer to Chapter 24.

Remark The target instruction of the branch is fetched at the EX stage of the branch instruction. Comparison of the operands of the branch instruction and calculation of the target address is performed at phase 2 of the RF stage and phase 1 of the EX stage of the instruction. Branch instructions require one cycle of the branch delay slot defined by the architecture. Jump instructions also require one cycle of delay slot. If the branch condition is not satisfied in a branch likely instruction, the instruction in its delay slot is nullified.

2.2.4 Special Instructions

Special instructions generate software exceptions. Their formats are R-type. For more information, refer to Chapter 24.

2.2.5 System Control Coprocessor (CP0) Instructions

System control coprocessor (CP0) instructions perform operations specifically on the CP0 registers to manipulate the memory management and exception handling facilities of the processor. Chapter 24 details CP0 instructions.

2.3 VR4101 CPU INSTRUCTION SET

Tables 2-5 to 2-19 list the instruction set architecture (IAS) and extended instruction set architecture (extended IAS) common to all VR-Series processors, extended instructions added in the VR4101, and CP0 instructions. These extended instructions help improve the performance of the OS by reducing the instruction code area. Multiprocessor instructions used in other VR-Series processors have been left out from the VR4101.

Table 2-5. CPU Instruction Set: Load and Store Instructions

Operation code	Description
LB	Load Byte
LBU	Load Byte Unsigned
LH	Load Halfword
LHU	Load Halfword Unsigned
LW	Load Word
LWL	Load Word Left
LWR	Load Word Right
SB	Store Byte
SH	Store Halfword
SW	Store Word
SWL	Store Word Left
SWR	Store Word Right

Table 2-6. CPU Instruction Set: Computational (Immediate) Instructions

Operation code	Description
ADDI	Add Immediate
ADDIU	Add Immediate Unsigned
SLTI	Set on Less Than Immediate
SLTIU	Set on Less Than Immediate Unsigned
ANDI	AND Immediate
ORI	OR Immediate
XORI	Exclusive OR Immediate
LUI	Load Upper Immediate

Table 2-7. CPU Instruction Set: Computational (3-Operand) Instructions

Operation code	Description
ADD	Add
ADDU	Add Unsigned
SUB	Subtract
SUBU	Subtract Unsigned
SLT	Set on Less Than
SLTU	Set on Less Than Unsigned
AND	AND
OR	OR
XOR	Exclusive OR
NOR	NOR

Table 2-8. CPU Instruction Set: Computational (Multiply and Divide) Instructions

Operation code	Description
MULT	Multiply
MULTU	Multiply Unsigned
DIV	Divide
DIVU	Divide Unsigned
MTHI	Move To HI
MTLO	Move To LO
MFHI	Move From HI
MFLO	Move From LO

Table 2-9. CPU Instruction Set: Jump and Branch Instructions

Operation code	Description
J	Jump
JAL	Jump And Link
JR	Jump Register
JALR	Jump And Link Register
BEQ	Branch on Equal
BNE	Branch on Not Equal
BLEZ	Branch on Less Than or Equal to Zero
BGTZ	Branch on Greater Than Zero
BLTZ	Branch on Less Than Zero
BGEZ	Branch on Greater Than or Equal to Zero
BLTZAL	Branch on Less Than Zero And Link
BGEZAL	Branch on Greater Than or Equal to Zero And Link
BC0T	Branch on Coprocessor 0 True
BC0F	Branch on Coprocessor 0 False

Table 2-10. CPU Instruction Set: Branch Likely Instructions

Operation code	Description
BEQL	Branch on Equal Likely
BNEL	Branch on Not Equal Likely
BLEZL	Branch on Less Than or Equal to Zero Likely
BGTZL	Branch on Greater Than Zero Likely
BLTZL	Branch on Less Than Zero Likely
BGEZL	Branch on Greater Than or Equal to Zero Likely
BLTZALL	Branch on Less Than Zero And Link Likely
BGEZALL	Branch on Greater Than or Equal to Zero And Link Likely
BC0TL	Branch on Coprocessor 0 True Likely
BC0FL	Branch on Coprocessor 0 False Likely

Table 2-11. CPU Instruction Set: Shift Instructions

Operation code	Description
SLL	Shift Left Logical
SRL	Shift Right Logical
SRA	Shift Right Arithmetic
SLLV	Shift Left Logical Variable
SRLV	Shift Right Logical Variable
SRAV	Shift Right Arithmetic Variable

Table 2-12. CPU Instruction Set: Special Instructions

Operation code	Description
SYNC	Synchronize memory references
SYSCALL	System Call
BREAK	Breakpoint
TGE	Trap if Greater Than or Equal
TGEU	Trap if Greater Than or Equal Unsigned
TLT	Trap if Less Than
TLTU	Trap if Less Than Unsigned
TEQ	Trap if Equal
TNE	Trap if Not Equal
TGEI	Trap if Greater Than or Equal Immediate
TGEIU	Trap if Greater Than or Equal Immediate Unsigned
TLTI	Trap if Less Than Immediate
TLTIU	Trap if Less Than Immediate Unsigned
TEI	Trap if Equal Immediate
TNEI	Trap if Not Equal Immediate

Table 2-13. CPU (Extended) Instructions: Load and Store Instructions

Operation code	Description
LD	Load Doubleword
LDL	Load Doubleword Left
LDR	Load Doubleword Right
LWU	Load Word Unsigned
SD	Store Doubleword
SDL	Store Doubleword Left
SDR	Store Doubleword Right

Table 2-14. CPU (Extended) Instructions: Computational (Immediate) Instructions

Operation code	Description
DADDI	Doubleword Add Immediate
DADDIU	Doubleword Add Immediate Unsigned

Table 2-15. CPU (Extended) Instructions: Computational (3-Operand) Instructions

Operation code	Description
DADD	Doubleword Add
DADDU	Doubleword Add Unsigned
DSUB	Doubleword Subtract
DSUBU	Doubleword Subtract Unsigned

Table 2-16. CPU (Extended) Instructions: Computational (Multiply and Divide) Instructions

Operation code	Description
DMULT	Doubleword Multiply
DMULTU	Doubleword Multiply Unsigned
DDIV	Doubleword Divide
DDIVU	Doubleword Divide Unsigned

Table 2-17. CPU (Extended) Instructions: Shift Instructions

Operation code	Description
DSLL	Doubleword Shift Left Logical
DSRL	Doubleword Shift Right Logical
DSRA	Doubleword Shift Right Arithmetic
DSLLV	Doubleword Shift Left Logical Variable
DSRLV	Doubleword Shift Right Logical Variable
DSRAV	Doubleword Shift Right Arithmetic Variable
DSLL32	Doubleword Shift Left Logical + 32
DSRL32	Doubleword Shift Right Logical +32
DSRA32	Doubleword Shift Right Arithmetic + 32

Table 2-18. CP0 Instructions

Operation code	Description
DMFC0	Doubleword Move From CP0
DMTC0	Doubleword Move To CP0
MTC0	Move to CP0
MFC0	Move from CP0
TLBR	Read Indexed TLB Entry
TLBWI	Write Indexed TLB Entry
TLBWR	Write Random TLB Entry
TLBP	Probe TLB for Matching Entry
ERET	Exception Return
CACHE	Cache Operation
HIBERNATE	Hibernate
SUSPEND	Suspend
STANDBY	Standby

Table 2-19. VR4101 Extended Instructions

Operation code	Description
MADD16	Multiply and Add 16bit
DMADD16	Doubleword Multiply and Add 16bit

[MEMO]

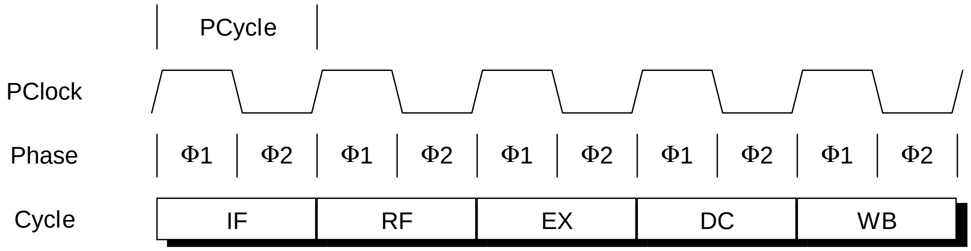
CHAPTER 3 VR4101 PIPELINE

This chapter describes the basic operation of the VR4101 processor pipeline, which includes descriptions of the delay slots (instructions that follow a branch or load instruction in the pipeline), interrupts to the pipeline flow caused by interlocks and exceptions, and CP0 hazards.

3.1 PIPELINE STAGES

The VR4101 has a five-stage instruction pipeline; each stage takes one PCycle (one cycle of PClock, which runs at quadruple of the frequency of MasterClock), and each PCycle has two phases: Φ1 and Φ2, as shown in Figure 3-1. Thus, the execution of each instruction takes at least 5 PCycles. An instruction can take longer - for example, if the required data is not in the cache, the data must be retrieved from main memory.

Figure 3-1. Pipeline Stages

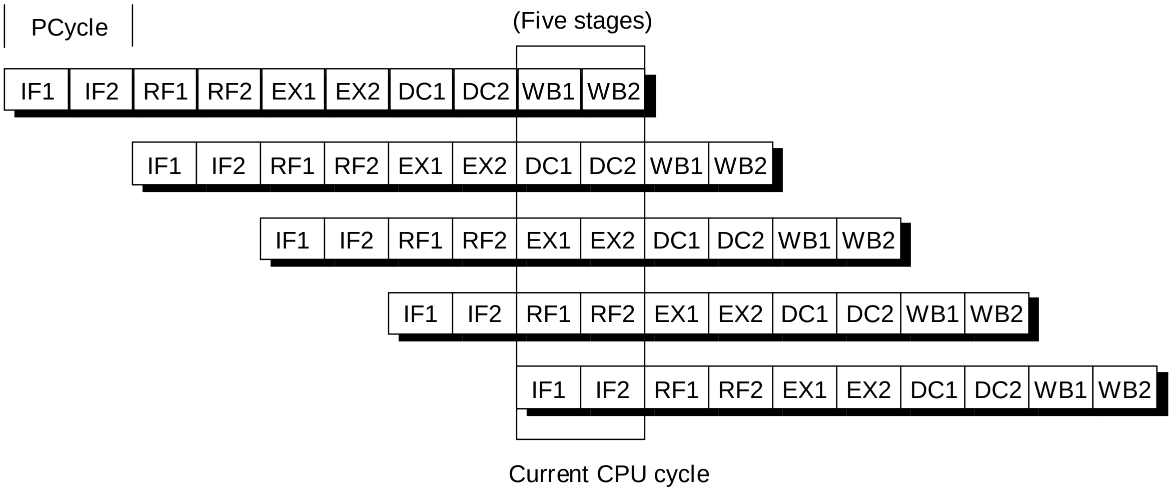


The five pipeline stages are:

- ✧ IF - Instruction cache fetch
- ✧ RF - Register fetch
- ✧ EX - Execution
- ✧ DC - Data cache fetch
- ✧ WB - Write back

Once the pipeline has been filled, five instructions are executed simultaneously. Figure 3-2 shows the five stages of the instruction pipeline; the next section describes the pipeline stages.

Figure 3-2. Instruction Execution in the Pipeline



3.1.1 Pipeline Activities

Figure 3-3 shows the activities that can occur during each pipeline stage; Table 3-1 describes these pipeline activities.

Figure 3-3. Pipeline Activities

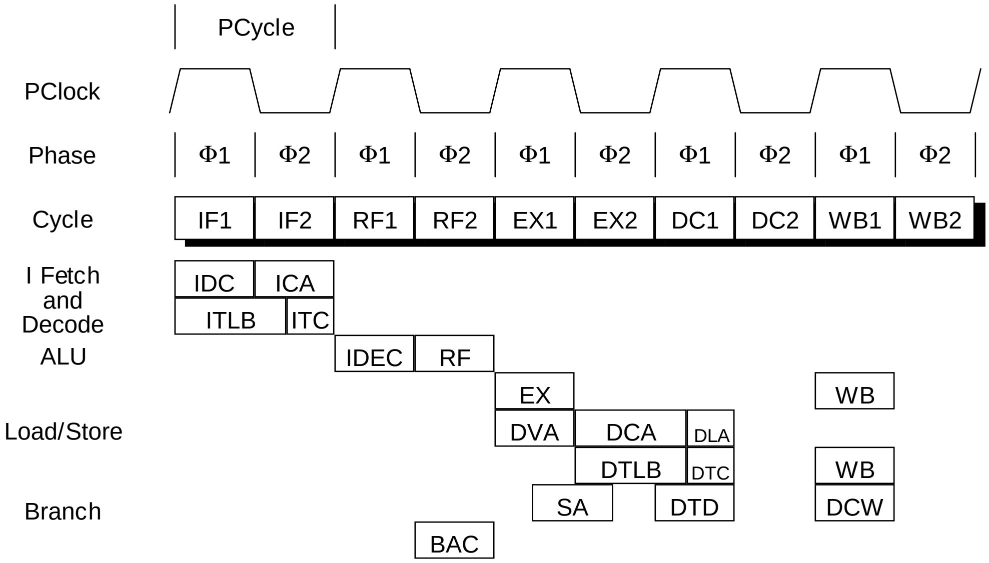


Table 3-1. Description of Pipeline Activities during Each Stage

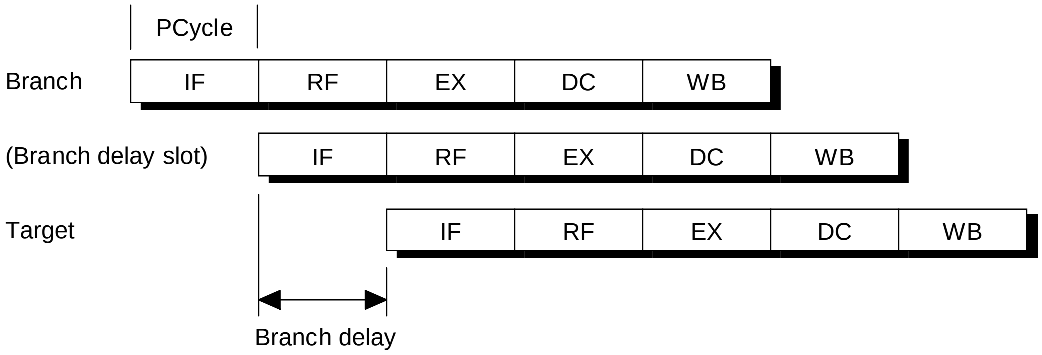
Cycle	Phase	Mnemonic	Description
IF	Φ1	ICD	Instruction cache address decode
		ITLB	Instruction address translation
	Φ2	ICA	Instruction cache array access
		ITC	Instruction tag check
RF	Φ1	IDEC	Instruction decode
	Φ2	RF	Register operand fetch
		BAC	Branch address calculation
EX	Φ1	EX	Execution stage
		DVA	Data virtual address calculation
		SA	Store align
	Φ2	DCA	Data cache address decode/array access
		DTLB	Data address translation
DC	Φ1	DLA	Data cache load align
		DTC	Data tag check
		DTD	Data transfer to data cache
WB	Φ1	DCW	Data cache write
		WB	Write back to register file

3.2 BRANCH DELAY

The VR4101 pipeline has a branch delay of one cycle, as a result of the branch comparison logic operating during the RF pipeline stage of the branch, producing an instruction address that is available in the IF stage, two instructions later.

Figure 3-4 illustrates the branch delay and the location of the branch delay slot.

Figure 3-4. Branch Delay



3.3 LOAD DELAY

A load instruction that does not allow its result to be used by the instruction immediately following is called a delayed load instruction. The instruction immediately following this delayed load instruction is referred to as the load delay slot.

In the VR4101, the instruction immediately following a load instruction can use the contents of the loaded register, however in such cases hardware interlocks insert additional delay cycles. Consequently, scheduling load delay slots can be desirable, both for performance and VR-Series processor compatibility.

3.4 PIPELINE OPERATION

The operation of the pipeline is illustrated by the following examples that describe how typical instructions are executed. The instructions described are: ADD, JALR, BEQ, TLT, LW, and SW. Each instruction is taken through the pipeline and the operations that occur in each relevant stage are described.

3.4.1 Add Instruction (Add rd, rs, rt)

IF stage In Φ1 of the IF stage, the eleven least-significant bits of the virtual access are used to access the instruction cache. In Φ2 of the IF stage, the cache index is compared with the page frame number and the cache data is read out. The virtual PC is incremented by 4 so that the next instruction can be fetched.

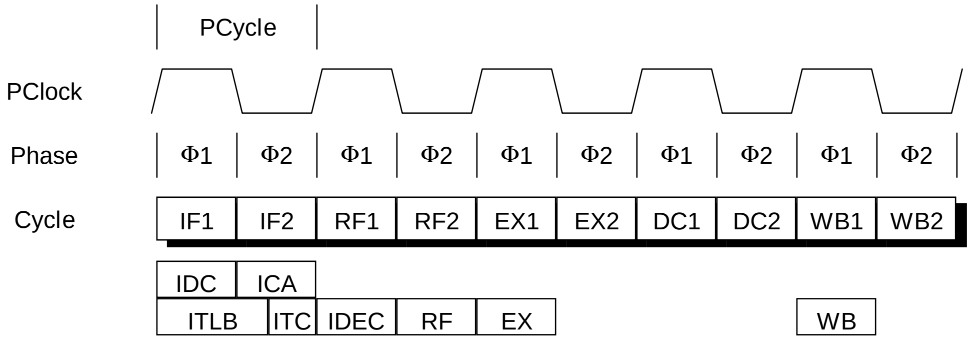
RF stage During Φ2, the 2-port register file is addressed with the rs and rt fields and the register data is valid at the register file output. At the same time, bypass multiplexers select inputs from either the EX- or DC-stage output in addition to the register file output, depending on the need for an operand bypass.

EX stage The ALU controls are set to do an A + B operation. The operands flow into the ALU inputs, and the ALU operation is started. The result of the ALU operation is latched into the ALU output latch during Φ1.

DC stage This stage is a NOP for this instruction. The data from the output of the EX stage (the ALU) is moved into the output latch of the DC.

WB stage During Φ1, the WB latch feeds the data to the inputs of the register file, which is addressed by the rd field. The file write strobe is enabled. By the end of Φ1, the data is written into the file.

Figure 3-5. Add Instruction Pipeline Activities



3.4.2 Jump and Link Register Instruction (JALR rd, rs)

IF stage Same as the IF stage for the ADD instruction.

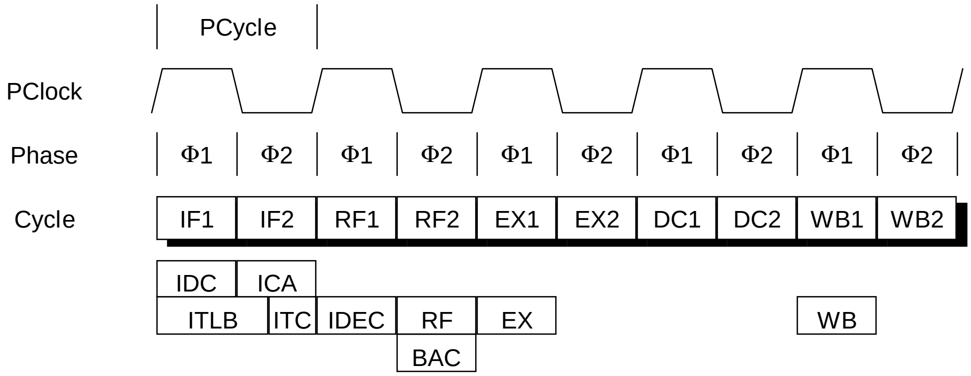
RF stage A register specified in the rs field is read from the file during Φ2 at the RF stage, and the value read from the rs register is input to the virtual PC latch synchronously. This value is used to fetch an instruction at the jump destination. The value of the virtual PC incremented during the IF stage is incremented again to produce the link address PC + 8 where PC is the address of the JALR instruction. The resulting value is the PC to which the program will eventually return. This value is placed in the Link output latch of the Instruction Address unit.

EX stage The PC + 8 value is moved from the Link output latch to the output latch of the EX stage.

DC stage The PC + 8 value is moved from the output latch of the EX stage to the output latch of the DC stage.

WB stage Refer to the ADD instruction. Note that if no value is explicitly provided for rd then register 31 is used as the default. If rd is explicitly specified, it cannot be the same register addressed by rs; if it is, the result of executing such an instruction is undefined.

Figure 3-6. JALR Instruction Pipeline Activities



3.4.3 Branch on Equal Instruction (BEQ rs, rt, offset)

IF stage — Same as the IF stage for the ADD instruction.

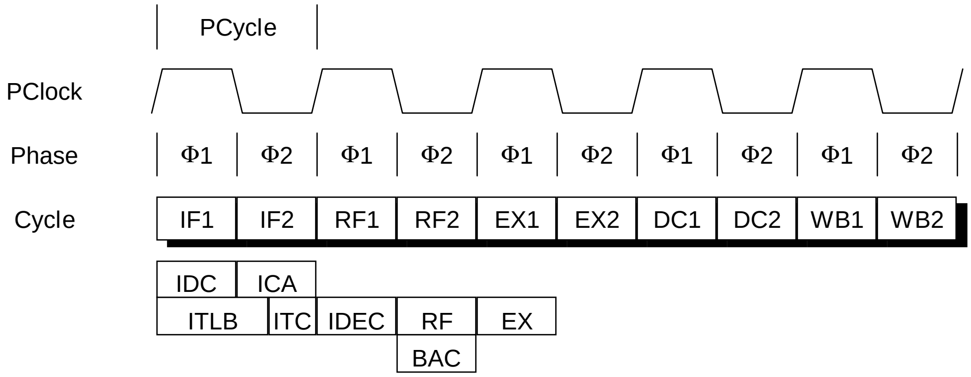
RF stage — During Φ2, the register file is addressed with the rs and rt fields. A check is performed to determine if each corresponding bit position of these two operands has equal values. If they are equal, the PC is set to PC + target, where target is the sign-extended offset field. If they are not equal, the PC is set to PC + 4.

EX stage — The next PC resulting from the branch comparison is valid at the beginning of Φ2 for instruction fetch.

DC stage — This stage is a NOP for this instruction.

WB stage — This stage is a NOP for this instruction.

Figure 3-7. BEQ Instruction Pipeline Activities



3.4.4 Trap if Less Than Instruction (TLT rs, rt)

IF stage — Same as the IF stage for the ADD instruction.

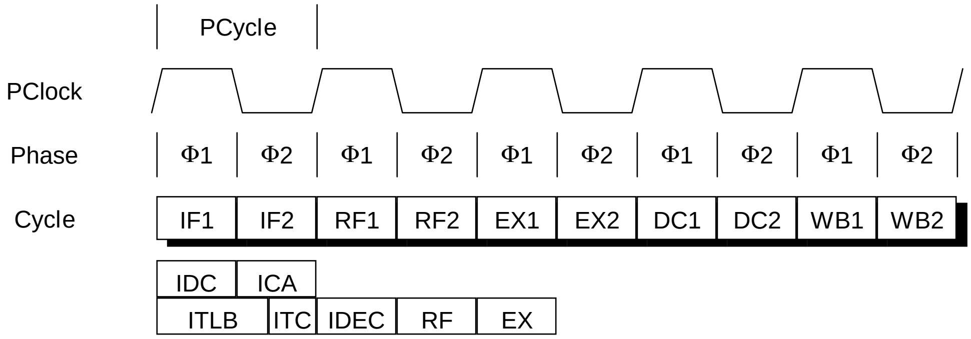
RF stage — Same as the RF stage for the ADD instruction.

EX stage — ALU controls are set to do an A - B operation. The operands flow into the ALU inputs, and the ALU operation is started. The result of the ALU operation is latched into the ALU output latch during Φ1. The sign bits of operands and of the ALU output latch are checked to determine if a less than condition is true. If this condition is true, a Trap exception occurs. The value in the PC register is used as an exception vector value, and from now on any instruction will be invalid.

DC stage — No operation

WB stage — The EPC register is loaded with the value of the PC if the less than condition was met in the EX stage. The Cause register ExCode field and BD bit are updated appropriately, as is the EXL bit of the Status register. If the less than condition was not met in the EX stage, no activity occurs in the WB stage.

Figure 3-8. TLT Instruction Pipeline Activities



3.4.5 Load Word Instruction (LW rt, offset (base))

IF stage — Same as the IF stage for the ADD instruction.

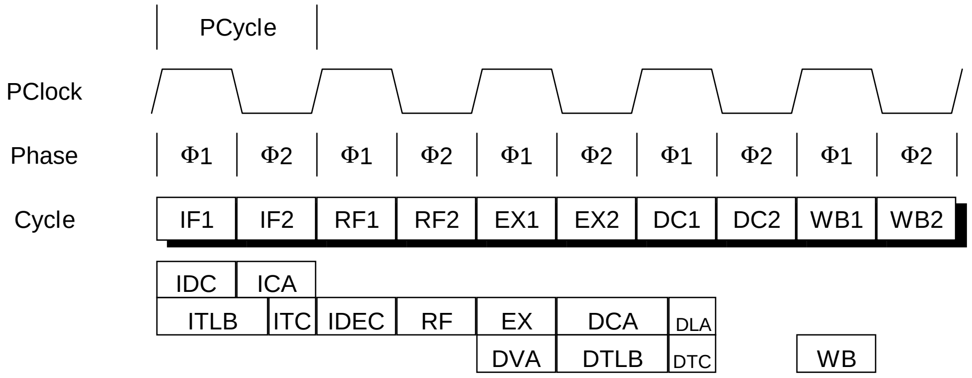
RF stage — Same as the RF stage for the ADD instruction. Note that the base field is in the same position as the rs field.

EX stage — Refer to the EX stage for the ADD instruction. For LW, the inputs to the ALU come from GPR[base] through the bypass multiplexer and from the sign-extended offset field. The result of the ALU operation that is latched into the ALU output latch in Φ1 represents the effective virtual address of the operand (DVA).

DC stage — The cache tag field is compared with the Page Frame Number (PFN) field of the TLB entry. After passing through the load aligner, aligned data is placed in the DC output latch during Φ2.

WB stage — During Φ1, the cache read data is written into the register file addressed by the rt field.

Figure 3-9. LW Instruction Pipeline Activities



3.4.6 Store Word Instruction (SW rt, offset (base))

IF stage Same as the IF stage for the ADD instruction.

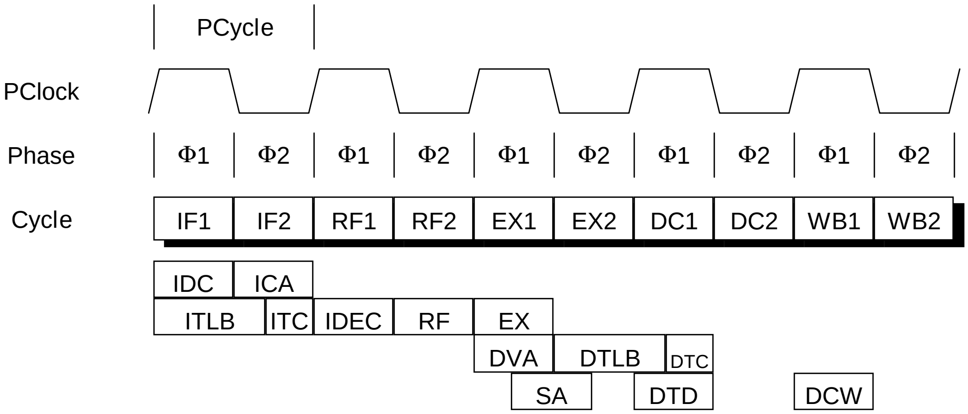
RF stage Same as the RF stage for the LW instruction.

EX stage Refer to the LW instruction for a calculation of the effective address. From the RF output latch, the GPR[rt] is sent through the bypass multiplexer and into the main shifter, where the shifter performs the byte-alignment operation for the operand. The results of the ALU are latched in the output latches during Φ1. The shift operations are latched in the output latches during Φ2.

DC stage Refer to the LW instruction for a description of the cache access.

WB stage If there was a cache hit, the content of the store data output latch is written into the data cache at the appropriate word location.
Note that all store instructions use the data cache for two consecutive PCycles. If the following instruction requires use of the data cache, the pipeline is slipped for one PCycle to complete the writing of an aligned store data.

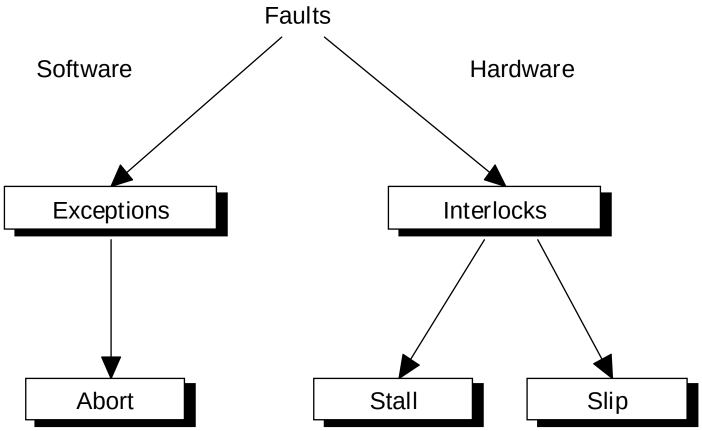
Figure 3-10. SW Instruction Pipeline Activities



3.5 INTERLOCK AND EXCEPTION HANDLING

Smooth pipeline flow is interrupted when cache misses or exceptions occur, or when data dependencies are detected. Interruptions handled using hardware, such as cache misses, are referred to as interlocks, while those that are handled using software are called exceptions. As shown in Figure 3-11, all interlock and exception conditions are collectively referred to as faults.

Figure 3-11. Interlocks, Exceptions, and Faults

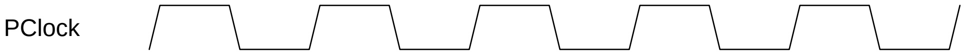


At each cycle, exception and interlock conditions are checked for all active instructions.

Because each exception or interlock condition corresponds to a particular pipeline stage, a condition can be traced back to the particular instruction in the exception/interlock stage, as shown in Figure 3-12. For instance, an LDI Interlock is raised in the Register Fetch (RF) stage.

Tables 3-2 to 3-4 describe the pipeline interlocks and exceptions listed in Figure 3-12.

Figure 3-12. Correspondence of Pipeline Stage to Interlock and Exception Condition



Phase	Φ1	Φ2	Φ1	Φ2	Φ1	Φ2	Φ1	Φ2	Φ1	Φ2

Stall

IF	RF	EX	DC	WB
	ITM		DTM	
	ICM		DCM	
			DCB	

Slip

IF	RF	EX	DC	WB
	LDI			
	MDI			
	SLI			
	CP0			

Exception

IF	RF	EX	DC	WB
IAErr	NMI	Trap	Reset	
	ITLB	OVF	DTLB	
	IPErr	DAErr	TMod	
	INTr		DPErr	
	IBE		WAT	
	SYSC		DBE	
	BP			
	CUn			
	RSVD			

Table 3-2. Description of Pipeline Stall

Stall	Description
ITM	Instruction TLB Miss
ICM	Instruction Cache Miss
DTM	Data TLB Miss
DCM	Data Cache Miss
DCB	Data Cache Busy

Table 3-3. Description of Pipeline Slip

Slip	Description
LDI	Load Data Interlock
MDI	MD Busy Interlock
SLI	Store-Load Interlock
CP0	Coprocessor 0 Interlock

Table 3-4. Description of Pipeline Exception

Exception	Description
IAErr	Instruction Address Error exception
NMI	Non-maskable Interrupt exception
ITLB	ITLB exception
IPErr	Instruction Parity Error exception
INTr	Interrupt exception
IBE	Instruction Bus Error exception
SYSC	System Call exception
BP	Breakpoint exception
CUn	Coprocessor Unusable exception
RSVD	Reserved Instruction exception
Trap	Trap exception
OVF	Overflow exception
DAErr	Data Address Error exception
Reset	Reset exception
DTLB	DTLB exception
DTMod	DTLB Modified exception
DPErr	Data Parity Error exception
WAT	Watch exception
DBE	Data Bus Error exception

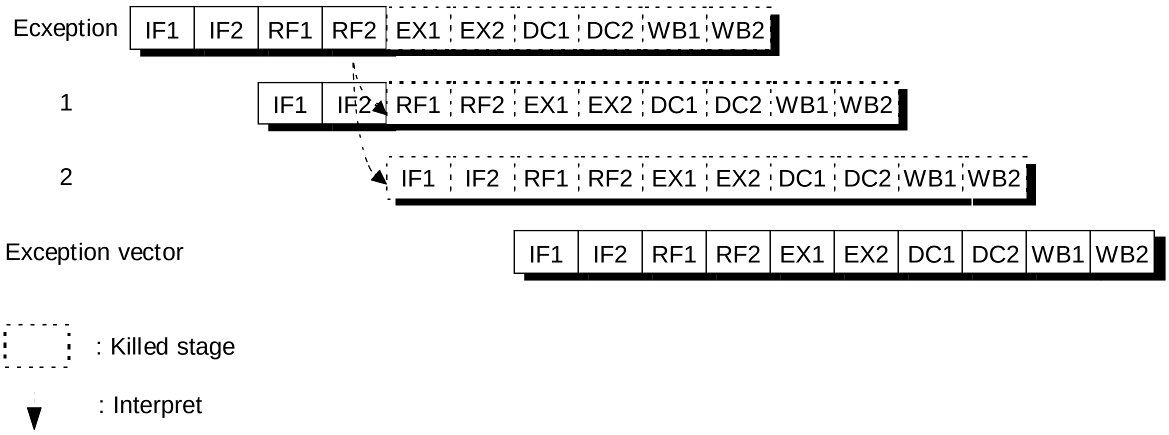
3.5.1 Exception Conditions

When an exception condition occurs, the relevant instruction and all those that follow it in the pipeline are cancelled. Accordingly, any stall conditions and any later exception conditions that may have referenced this instruction are inhibited; there is no benefit in servicing stalls for a cancelled instruction.

When an exceptional conditions is detected for an instruction, the VR4101 will kill it and all following instructions. When this instruction reaches the WB stage, the exception flag and various information items are written to CP0 registers. The current PC is changed to the appropriate exception vector address and the exception bits of earlier pipeline stages are cleared.

This implementation allows all preceding instructions to complete execution and prevents all subsequent instructions from completing. Thus the value in the EPC is sufficient to restart execution. It also ensures that exceptions are taken in the order of execution; an instruction taking an exception may itself be killed by an instruction further down the pipeline that takes an exception in a later cycle.

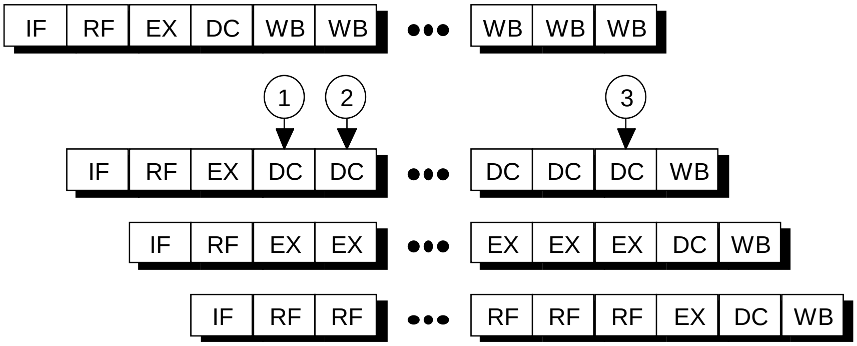
Figure 3-13. Exception Detection



3.5.2 Stall Conditions

Stalls are used to stop the pipeline for conditions detected after the RF stage. When a stall occurs, the processor will resolve the condition and then the pipeline will continue. Figure 3-14 shows a data cache miss stall, and Figure 3-15 shows a CACHE instruction stall.

Figure 3-14. Data Cache Miss Stall


IF RF EX DC WB WB ●●● WB WB WB
1
2
3
IF RF EX DC DC ●●● DC DC DC WB
IF RF EX EX ●●● EX EX EX DC WB
IF RF RF ●●● RF RF RF EX DC WB

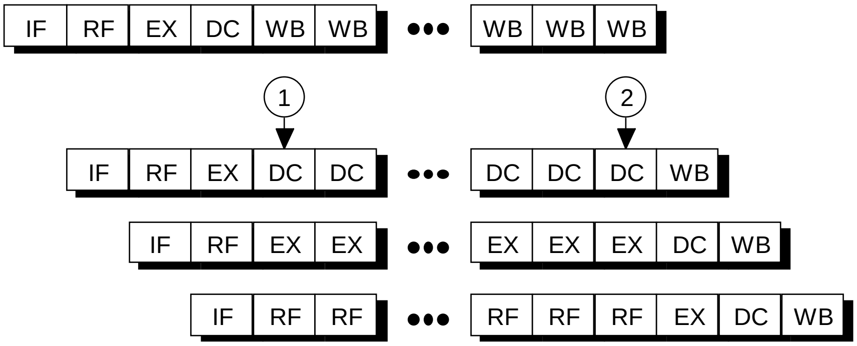
① Detect data cache miss

② Start moving data cache line to write buffer

③ Get last word into cache and restart pipeline

If the cache line to be replaced is dirty — the W bit is set — the data is moved to the internal write buffer in the next cycle. The write-back data is returned to memory. The last word in the data is returned to the cache at 3, and pipelining restarts.

Figure 3-15. CACHE Instruction Stall


IF RF EX DC WB WB ●●● WB WB WB
1
2
IF RF EX DC DC ●●● DC DC DC WB
IF RF EX EX ●●● EX EX EX DC WB
IF RF RF ●●● RF RF RF EX DC WB

① CACHE instruction start

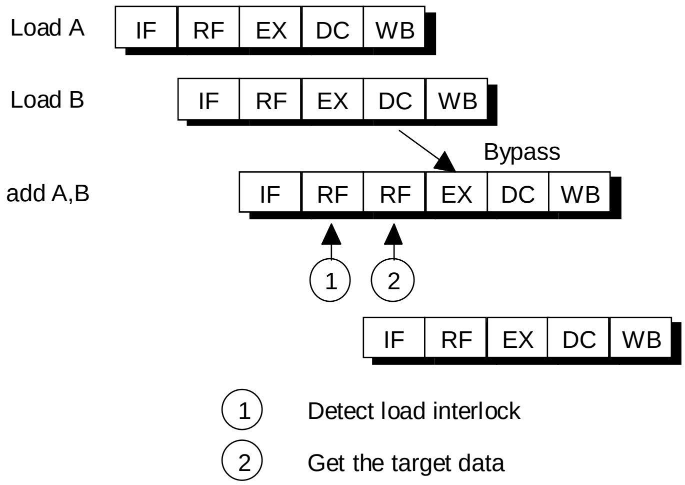
② CACHE instruction complete

When the CACHE instruction enters the DC pipe-stage, the pipeline stalls while the CACHE instruction is executed. The pipeline begins running again when the CACHE instruction is completed, allowing the instruction fetch to proceed.

3.5.3 Slip Conditions

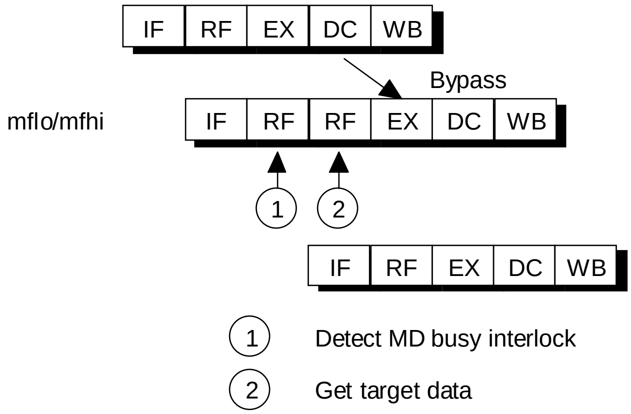
During Φ2 of the RF stage and Φ1 of the EX stage, internal logic will determine whether it is possible to start the current instruction in this cycle. If all of the source operands are available (either from the register file or via the internal bypass logic) and all the hardware resources necessary to complete the instruction will be available whenever required, then the instruction “run”; otherwise, the instruction will “slip”. Slipped instructions are retired on subsequent cycles until they issue. The backend of the pipeline (stages DC and WB) will advance normally during slips in an attempt to resolve the conflict. NOPs will be inserted into the bubble in the pipeline. Instructions killed by branch likely instructions, ERET or exceptions will not cause slips.

Figure 3-16. Load Data Interlock



Load Data Interlock is detected in the RF stage shown in as Figure 3-16 and also the pipeline slips in the stage. Load Data Interlock occurs when data fetched by a load instruction and data moved from HI, LO or CP0 register is required by the next immediate instruction. The pipeline begins running again when the clock after the target of the load is read from the data cache, HI, LO and CP0 register. The data returned at the end of the DC stage is input into the end of the RF stage, using the bypass multiplexers.

Figure 3-17. MD Busy Interlock



MD Busy Interlock is detected in the RF stage as shown in Figure 3-17 and also the pipeline slips in the stage. MD Busy Interlock occurs when Hi/Lo register is required by MFHi/Lo instruction before finishing Mult/Div execution. The pipeline begins running again the clock after finishing Mult/Div execution. The data returned from the Hi/Lo register at the end of the DC stage is input into the end of the RF stage, using the bypass multiplexers.

Store-Load Interlock is detected in the EX stage and the pipeline slips in the RF stage. Store-Load Interlock occurs when store instruction followed by load instruction is detected. The pipeline begins running again one clock after.

Coprocessor 0 Interlock is detected in the EX stage and the pipeline slips in the RF stage. A coprocessor interlock occurs when an MTC0 instruction for the Configuration or Status register is detected.

The pipeline begins running again one clock after.

3.5.4 Bypassing

In some cases, data and conditions produced in the EX, DC and WB stages of the pipeline are made available to the EX stage (only) through the bypass datapath.

Operand bypass allows an instruction in the EX stage to continue without having to wait for data or conditions to be written to the register file at the end of the WB stage. Instead, the Bypass Control Unit is responsible for ensuring data and conditions from later pipeline stages are available at the appropriate time for instructions earlier in the pipeline.

The Bypass Control Unit is also responsible for controlling the source and destination register addresses supplied to the register file.

3.6 CODE COMPATIBILITY

The VR4101 can execute all programs that can be executed in other VR-Series processors. But the reverse is not necessarily true. Programs complied using a standard MIPS compiler can be executed in both types of processors. When using manual assembly, however, write programs carefully so that compatibility with other VR series processors can be maintained. Matters which should be paid attention to when porting programs between the VR4101 and other VR-Series processors are listed below.

- The VR4100 CPU core does not support floating-point instructions since it has no Floating-Point Unit (FPU).
- Multiply-add instructions (DMADD16, MADD16) are added in the VR4100 CPU core.
- Instructions for power modes (HIBERNATE, STANDBY, SUSPEND) are added in the VR4100 CPU core to support power modes.
- The VR4100 CPU core does not have the LL bit to perform synchronization of multiprocessing. Therefore, the CPU core does not support instructions which manipulate the LL bit (LL, LLD, SC, SCD).

For more information, refer to Chapter 24, *the VR4000, VR4400 User’s Manual*, or *the VR4200 User’s Manual*.

[MEMO]

CHAPTER 4 MEMORY MANAGEMENT SYSTEM

The VR4101 provides a memory management unit (MMU) which uses a translation lookaside buffer (TLB) to translate virtual addresses into physical addresses. This chapter describes the virtual and physical address spaces, the virtual-to-physical address translation, the operation of the TLB in making these translations, and the CP0 registers that provide the software interface to the TLB.

4.1 TRANSLATION LOOKASIDE BUFFER (TLB)

Virtual addresses are translated into physical addresses using an on-chip TLB [Note]. The on-chip TLB is a fully-associative memory that holds 32 entries, which provide mapping to 32 odd/even page pairs for one entry. The pages can have five different sizes, 1 K, 4 K, 16 K, 64 K, and 256 K. If it is supplied with a virtual address, each of the 32 TLB entries is checked simultaneously to see whether they match the virtual addresses that are provided with the ASID field and saved in the EntryHi register.

Note Virtual addresses may be converted to physical addresses without using a TLB, depending on the address space that is being subjected to address translation. For example, address translation for the kseg0 or kseg1 address space does not use mapping. The physical addresses of these address spaces are determined by subtracting the base address of the address space from the virtual addresses.

4.1.1 Hits and Misses

If there is a virtual address match, or “hit,” in the TLB, the physical page number is extracted from the TLB and concatenated with the offset to form the physical address.

If no match occurs (TLB “miss”), an exception is taken and software refills the TLB from the page table resident in memory. The software writes to an entry selected using the Index register or a random entry indicated in the Random register.

4.1.2 Multiple Hit

If more than one entry in the TLB matches the virtual address being translated, the operation is undefined and the TLB may be disabled. In this case, the TLB-Shutdown (TS) bit of the Status register is set to 1, and the TLB becomes unusable (an attempt to access the TLB results in a TLB Mismatch exception regardless of whether there is an entry that hits). The TS bit can be cleared only by a reset.

4.2 ADDRESS SPACES

This section describes the virtual and physical address spaces and the manner in which virtual addresses are converted or “translated” into physical addresses in the TLB.

4.2.1 Virtual Address Space

The VR4101 virtual address can be either 32 or 64 bits wide, depending on whether the processor is operating in 32-bit or 64-bit mode.

- ✧ In 32-bit mode, addresses are 32 bits wide. The maximum user process size is 2 Gbytes (2^{31}).
- ✧ In 64-bit mode, addresses are 64 bits wide. The maximum user process size is 1 Tbyte (2^{40}).

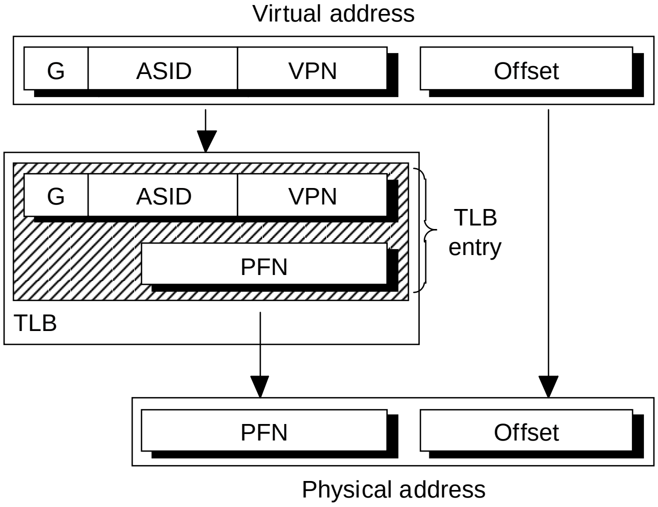
Figure 4-1 shows the translation of a virtual address into a physical address.

Figure 4-1. Virtual-to-Physical Address Translation

1 The virtual page number (VPN) in the virtual address (VA) is compared with the VPN in the TLB.

2 If there is a match, the page frame number (PFN) representing the high-order bits of the physical address is output from the TLB.

3 The offset is then added to the PFN passing through the TLB.



As shown in Figures 4-2 and 4-3, the virtual address is extended with an address space identifier (ASID), which reduces the frequency of TLB flushing when switching contexts. This 8-bit ASID is in the CP0 EntryHi register, described later in this chapter. The Global (G) bit is in the EntryLo0 and EntryLo1 registers, described later in this chapter.

4.2.2 Physical Address Space

Using a 32-bit address, the processor physical address space encompasses 4 Gbytes. The VR4101 uses this 4-Gbyte physical address space as shown in Figure 4-2.

Figure 4-2. VR4101 Physical Address Space

0xFFFF FFFF 0x2000 0000	(Mirror image of 0x0000 0000 - 0x1FFF FFFF area)
0x1FFF FFFF 0x1F00 0000	ROM area (include boot ROM)
0x1EFF FFFF 0x1900 0000	RFU
0x18FF FFFF 0x1800 0000	(Mirror image of 0x1F00 0000 - 0x1FFF FFFF area)
0x17FF FFFF 0x1700 0000	ISA I/O area for 16-bit device (for PCMCIA)
0x16FF FFFF 0x1600 0000	ISA I/O area for 8-bit device (for PCMCIA)
0x15FF FFFF 0x1500 0000	ISA memory area for 16-bit device (for PCMCIA)
0x14FF FFFF 0x1400 0000	ISA memory area for 8-bit device (for PCMCIA)
0x13FF FFFF 0x0C00 0000	RFU
0x0BFF FFFF 0x0B00 0000	Hardware register area
0x0AFF FFFF 0x0A00 0000	LCD display buffer
0x09FF FFFF 0x0400 0000	RFU
0x03FF FFFF 0x0000 0000	DRAM area

The following section describes the translation of a virtual address to a physical address.

4.2.3 Virtual-to-Physical Address Translation

Converting a virtual address to a physical address begins by comparing the virtual address from the processor with the virtual addresses in the TLB; there is a match when the virtual page number (VPN) of the address is the same as the VPN field of the entry, and either:

- ✧ the Global (G) bit of the TLB entry is set to 1, or
- ✧ the ASID field of the virtual address is the same as the ASID field of the TLB entry.

This match is referred to as a TLB hit. If there is no match, a TLB Mismatch exception is taken by the processor and software is allowed to refill the TLB from a page table of virtual/physical addresses in memory.

If there is a virtual address match in the TLB, the physical address is output from the TLB and concatenated with the offset, which represents an address within the page frame space. The offset does not pass through the TLB. Instead, the low-order bits of the virtual address are output without being translated. See descriptions about the virtual address space for details.

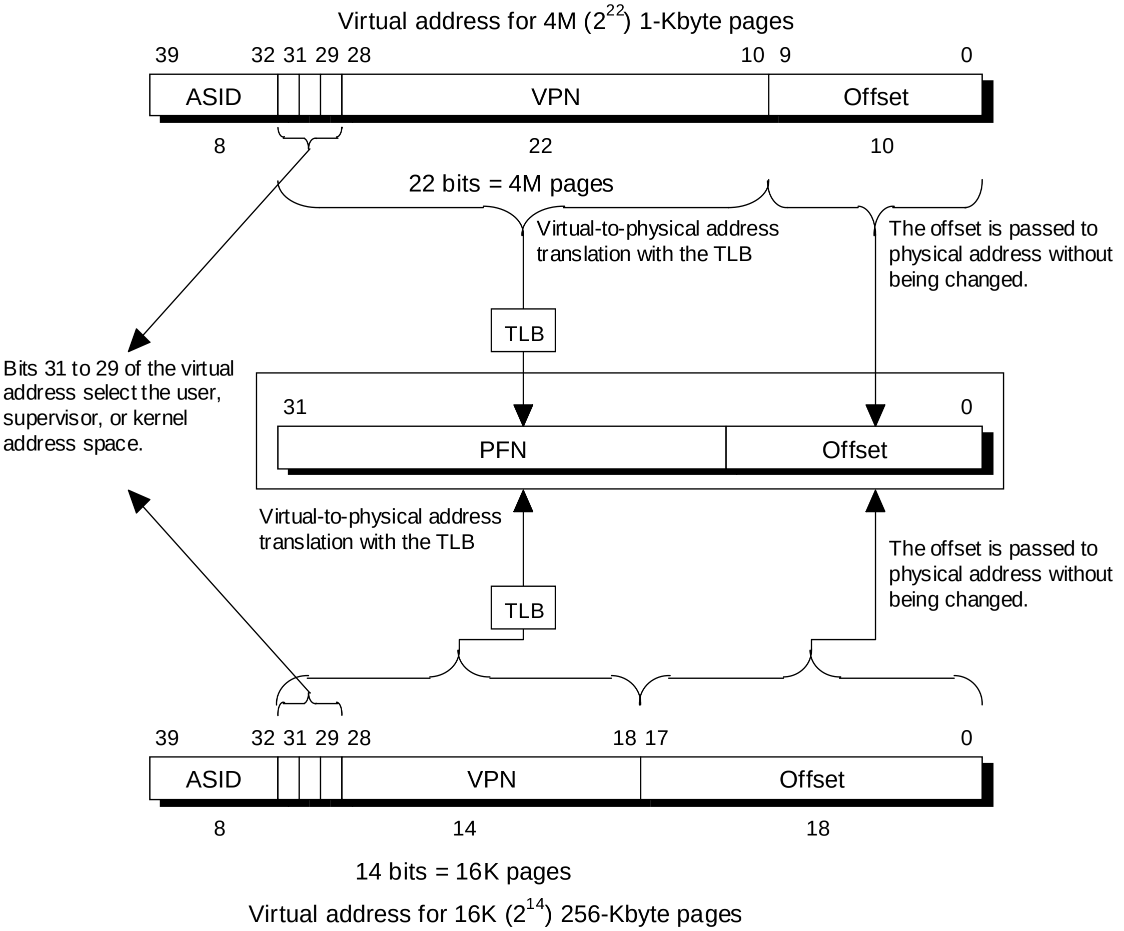
The next two sections describe the 32-bit and 64-bit mode address translations.

4.2.4 32-bit Mode Address Translation

Figure 4-3 shows the virtual-to-physical-address translation of a 32-bit mode address. The pages can have five different sizes between 1 Kbyte (10 bits) and 256 Kbytes (18 bits), each being 4 times as large as the preceding one in ascending order, that is 1 K, 4 K, 16 K, 64 K, and 256 K.

- ✧ Shown at the top of Figure 4-3 is the virtual address space in which the page size is 1 Kbyte and the offset is 10 bits. The 22 bits excluding the ASID field represents the virtual page number (VPN), enabling selecting a page table of 4 M entries.
- ✧ Shown at the bottom of Figure 4-3 is the virtual address space in which the page size is 256 Kbytes and the offset is 18 bits. The 14 bits excluding the ASID field represents the VPN, enabling selecting a page table of 16 K entries.

Figure 4-3. 32-bit Mode Virtual Address Translation

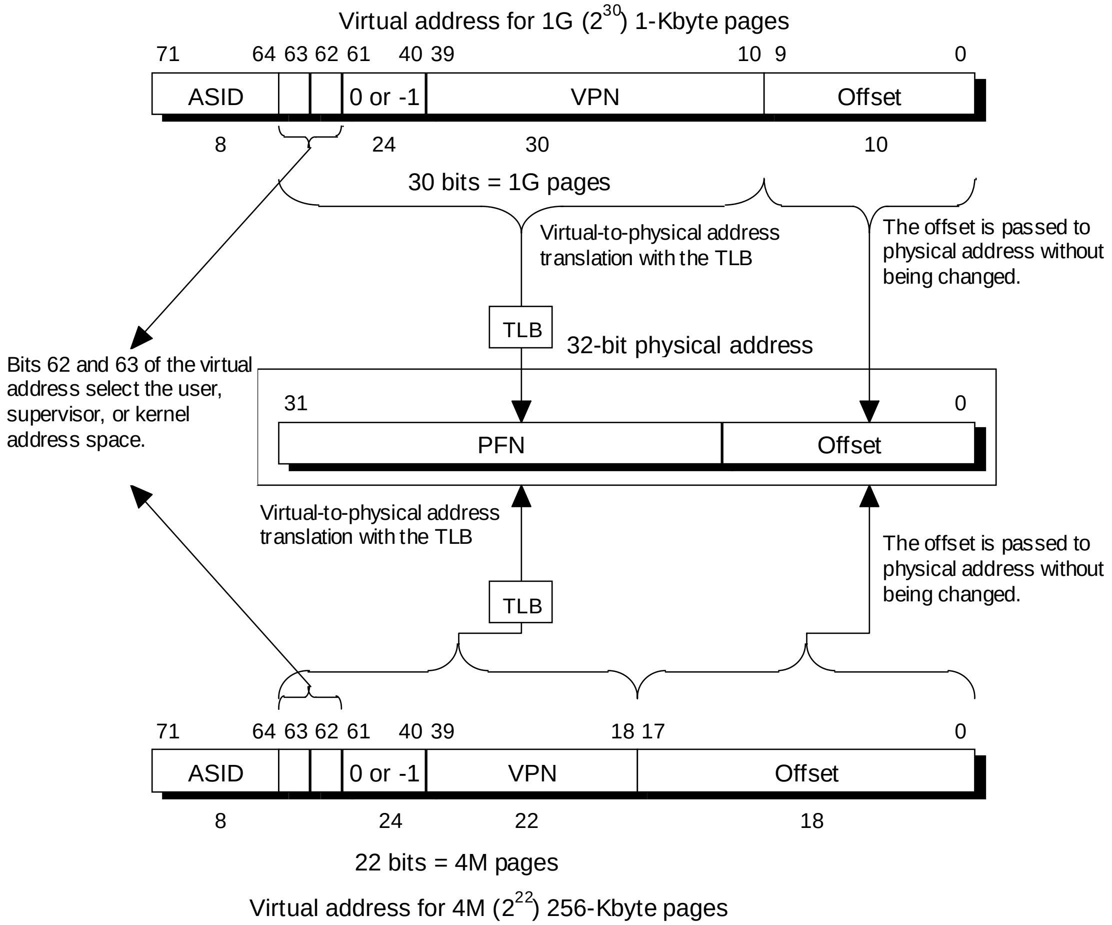

Virtual address for 4M (2^{22}) 1-Kbyte pages
39
32 31 29 28
10 9
0
ASID
VPN
Offset
8
22
10
22 bits = 4M pages
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
Bits 31 to 29 of the virtual address select the user, supervisor, or kernel address space.
31
0
PFN
Offset
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
39
32 31 29 28
18 17
0
ASID
VPN
Offset
8
14
18
14 bits = 16K pages
Virtual address for 16K (2^{14}) 256-Kbyte pages

4.2.5 64-bit Mode Address Translation

Figure 4-4 shows the virtual-to-physical-address translation of a 64-bit mode address. This figure illustrates the two possible page sizes: a 1-Kbyte page (10 bits) and a 256-Kbyte page (18 bits).

- ✧ Shown at the top of Figure 4-4 is the virtual address space in which the page size is 1 Kbyte and the offset is 10 bits. The 30 bits excluding the ASID field represents the virtual page number (VPN), enabling selecting a page table of 1 G entry.
- ✧ Shown at the bottom of Figure 4-4 is the virtual address space in which the page size is 256 Kbytes and the offset is 18 bits. The 22 bits excluding the ASID field represents the VPN, enabling selecting a page table of 4 M entries.

Figure 4-4. 64-bit Mode Virtual Address Translation


Virtual address for 1G (2^{30}) 1-Kbyte pages
71
64 63 62 61
40 39
10 9
0
ASID
0 or -1
VPN
Offset
8
24
30
10
30 bits = 1G pages
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
32-bit physical address
Bits 62 and 63 of the virtual address select the user, supervisor, or kernel address space.
31
0
PFN
Offset
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
71
64 63 62 61
40 39
18 17
0
ASID
0 or -1
VPN
Offset
8
24
22
18
22 bits = 4M pages
Virtual address for 4M (2^{22}) 256-Kbyte pages

4.2.6 Operating Modes

The processor has three operating modes that function in both 32- and 64-bit operations:

- ✧ User mode
- ✧ Supervisor mode
- ✧ Kernel mode

User and Kernel modes are common to all VR-Series processors. Generally, Kernel mode is used to executing the operating system, while User mode is used to run application programs. The VR4000 series processors have a third mode, which is called Supervisor mode and categorized in between User and Kernel modes. This mode is used to configure a high-security system.

When an exception occurs, the CPU enters Kernel mode, and remains in this mode until an exception return instruction (ERET) is executed. The ERET instruction brings back the processor to the mode in which it was just before the exception occurs.

These modes are described in the next three sections.

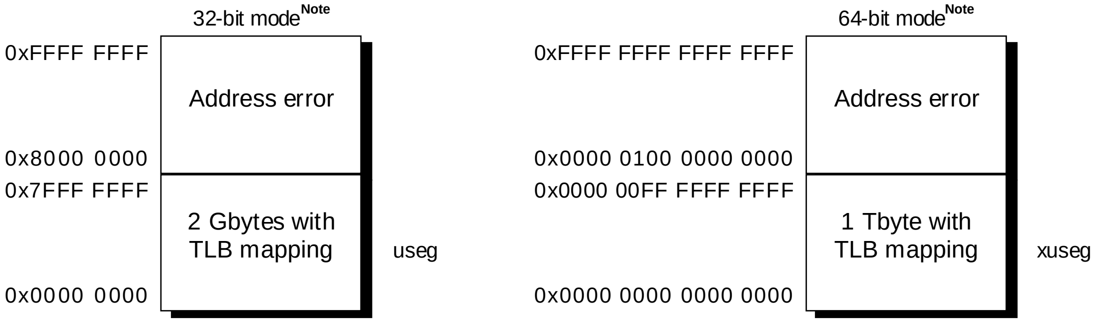
(1) User-mode virtual addressing

In User mode, a single virtual address space labeled User segment is available; its size is:

- ✧ 2 Gbytes (2^{31} bytes) in 32-bit mode (useg)
- ✧ 1 Tbyte (2^{40} bytes) in 64-bit mode (xuseg)

Table 4-1 lists the characteristics of each user segment (useg and xuseg).

Figure 4-5. User Mode Address Space



Note The VR4101 uses 64-bit addresses within it. When the processor is running in Kernel mode, it saves the contents of each register or restores their previous contents to initialize them before switching the context. For 32-bit mode addressing, bit 31 is sign-extended to bits 32 to 63, and the resulting 32 bits are used for addressing. Usually, it is impossible for 32-bit mode programs to generate invalid addresses. If context switching occurs and the processor enters Kernel mode, however, an attempt may be made to save an address other than the sign-extended 32-bit address mentioned above to a 64-bit register. In this case, user-mode programs are likely to generate an invalid address.

The User segment starts at address 0 and the current active user process resides in either useg (in 32-bit mode) or xuseg (in 64-bit mode). The TLB identically maps all references to useg/xuseg from all modes, and controls cache accessibility.

The processor operates in User mode when the Status register contains the following bit-values:

- KSU = 10
- EXL = 0
- ERL = 0

In conjunction with these bits, the UX bit in the Status register selects 32- or 64-bit User mode addressing as follows:

- When UX = 0, 32-bit useg space is selected.
- When UX = 1, 64-bit xuseg space is selected.

Table 4-1. Comparison of useg and xuseg

Address bit value	Status register bit value				Segment name	Address range	Size
	KSU	EXL	ERL	UX			
32-bit A[31] = 0	10	0	0	0	useg	0x0000 0000 to 0x7FFF FFFF	2 Gbytes (2^{31} bytes)
64-bit A[63..40] = 0	10	0	0	1	xuseg	0x0000 0000 0000 0000 to 0x0000 00FF FFFF FFFF	1 Tbyte (2^{40} bytes)

(a) useg (32-bit mode)

In User mode, when UX = 0 in the Status register, User mode addressing is compatible with the 32-bit addressing model shown in Figure 4-5, and a 2-Gbyte user address space is available, labelled useg.

All valid User mode virtual addresses have their most-significant bit cleared to 0; any attempt to reference an address with the most-significant bit set while in User mode causes an Address Error exception.

In 32-bit User mode addressing, the TLB Mismatch exception vector is used for TLB misses.

The system maps all references to useg through the TLB, and bit settings within the TLB entry for the page determine the cacheability of a reference.

(b) xuseg (64-bit mode)

In User mode, when UX =1 in the Status register, User mode addressing is extended to the 64-bit addressing model shown in Figure 4-5. In 64-bit User mode, the processor provides a single address space of 2^{40} bytes, labelled xuseg.

All valid User mode virtual addresses have bits 63:40 equal to 0; an attempt to reference an address with bits 63:40 equal to 1 causes an Address Error exception.

The XTLB Mismatch exception vector is used for TLB misses.

(2) Supervisor-mode virtual addressing

Supervisor mode is designed for layered operating systems in which a true kernel runs in Kernel mode, and the rest of the operating system runs in Supervisor mode.

The processor operates in Supervisor mode when the Status register contains the following bit-values:

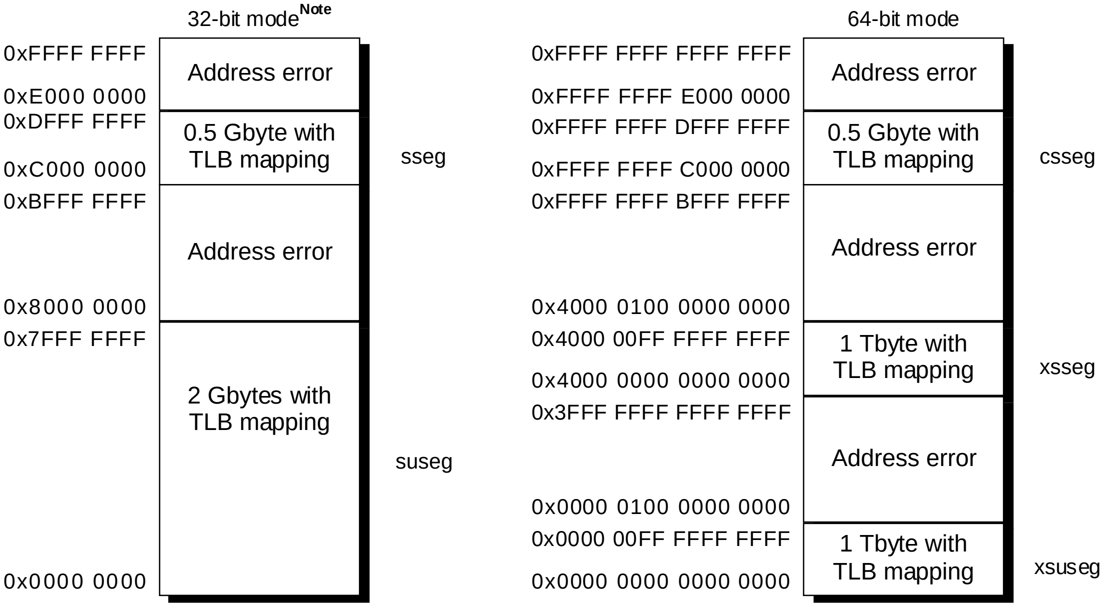
- KSU = 01
- EXL = 0
- ERL = 0

In conjunction with these bits, the SX bit in the Status register selects 32- or 64-bit Supervisor mode addressing:

- When SX = 0, 32-bit supervisor space is selected.
- When SX = 1, 64-bit supervisor space is selected.

Figure 4-6 shows Supervisor mode address mapping. Table 4-2 lists the characteristics of the Supervisor mode segments; descriptions of the address spaces follow.

Figure 4-6. Supervisor Mode Address Space



Note The VR4101 uses 64-bit addresses within it. For 32-bit mode addressing, bit 31 is sign-extended to bits 32 to 63, and the resulting 32 bits are used for addressing. Usually, it is impossible for 32-bit mode programs to generate invalid addresses. In an operation of base register + offset for addressing, however, a two's complement overflow may occur, causing an invalid address. Note that the result becomes undefined. Two factors that can cause a two's complement follow:

- When offset bit 15 is 0, base register bit 31 is 0, and bit 31 of the operation “base register + offset” is 1
- When offset bit 15 is 1, base register bit 31 is 1, and bit 31 of the operation “base register + offset” is 0

Table 4-2. 32-bit and 64-bit Supervisor Mode Segments

Address bit value	Status register bit value				Segment name	Address range	Size
	KSU	EXL	ERL	SX			
32-bit A[31] = 0	01	0	0	0	suseg	0x0000 0000 to 0x7FFF FFFF	2 Gbytes (2^{31} bytes)
32-bit A[31..29] = 110	01	0	0	0	sseg	0xC000 0000 to 0xDFFF FFFF	512 Mbytes (2^{29} bytes)
64-bit A[63..62] = 00	01	0	0	1	xsuseg	0x0000 0000 0000 0000 to 0x0000 00FF FFFF FFFF	1 Tbyte (2^{40} bytes)
64-bit A[63..62] = 01	01	0	0	1	xsseg	0x4000 0000 0000 0000 to 0x4000 00FF FFFF FFFF	1 Tbyte (2^{40} bytes)
64-bit A[63..62] = 11	01	0	0	1	csseg	0xFFFF FFFF C000 0000 to 0xFFFF FFFF DFFF FFFF	512 Mbytes (2^{29} bytes)

(a) suseg (32-bit Supervisor mode, user space)

When SX = 0 in the Status register and the most-significant bit of the virtual address space is set to 0, the suseg virtual address space is selected; it covers 2 Gbytes (2^{31} bytes) of the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space starts at virtual address 0x0000 0000 and runs through 0x7FFF FFFF.

(b) sseg (32-bit Supervisor mode, supervisor space)

When SX = 0 in the Status register and the three most-significant bits of the virtual address space are 110, the sseg virtual address space is selected; it covers 512 Mbytes (2^{29} bytes) of the current supervisor virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address 0xC000 0000 and runs through 0xDFFF FFFF.

(c) xsuseg (64-bit Supervisor mode, user space)

When SX = 1 in the Status register and bits 63 and 62 of the virtual address space are set to 00, the xsuseg virtual address space is selected; it covers 1 Tbyte (2^{40} bytes) of the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space starts at virtual address 0x0000 0000 0000 0000 and runs through 0x0000 00FF FFFF FFFF.

(d) xsseg (64-bit Supervisor mode, current supervisor space)

When SX = 1 in the Status register and bits 63 and 62 of the virtual address space are set to 01, the xsseg virtual address space is selected; it covers 1 Tbyte (2^{40} bytes) of the current supervisor virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address 0x4000 0000 0000 0000 and runs through 0x4000 00FF FFFF FFFF.

(e) csseg (64-bit Supervisor mode, separate supervisor space)

When SX = 1 in the Status register and bits 63 and 62 of the virtual address space are set to 11, the csseg virtual address space is selected; it covers 512 Mbytes (2^{29} bytes) of the separate supervisor virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address 0xFFFF FFFF C000 0000 and runs through 0xFFFF FFFF DFFF FFFF.

(3) Kernel-mode virtual addressing

If the Status register satisfies any of the following conditions, the processor runs in Kernel mode.

- ✧ KSU = 00
- ✧ EXL = 1
- ✧ ERL = 1

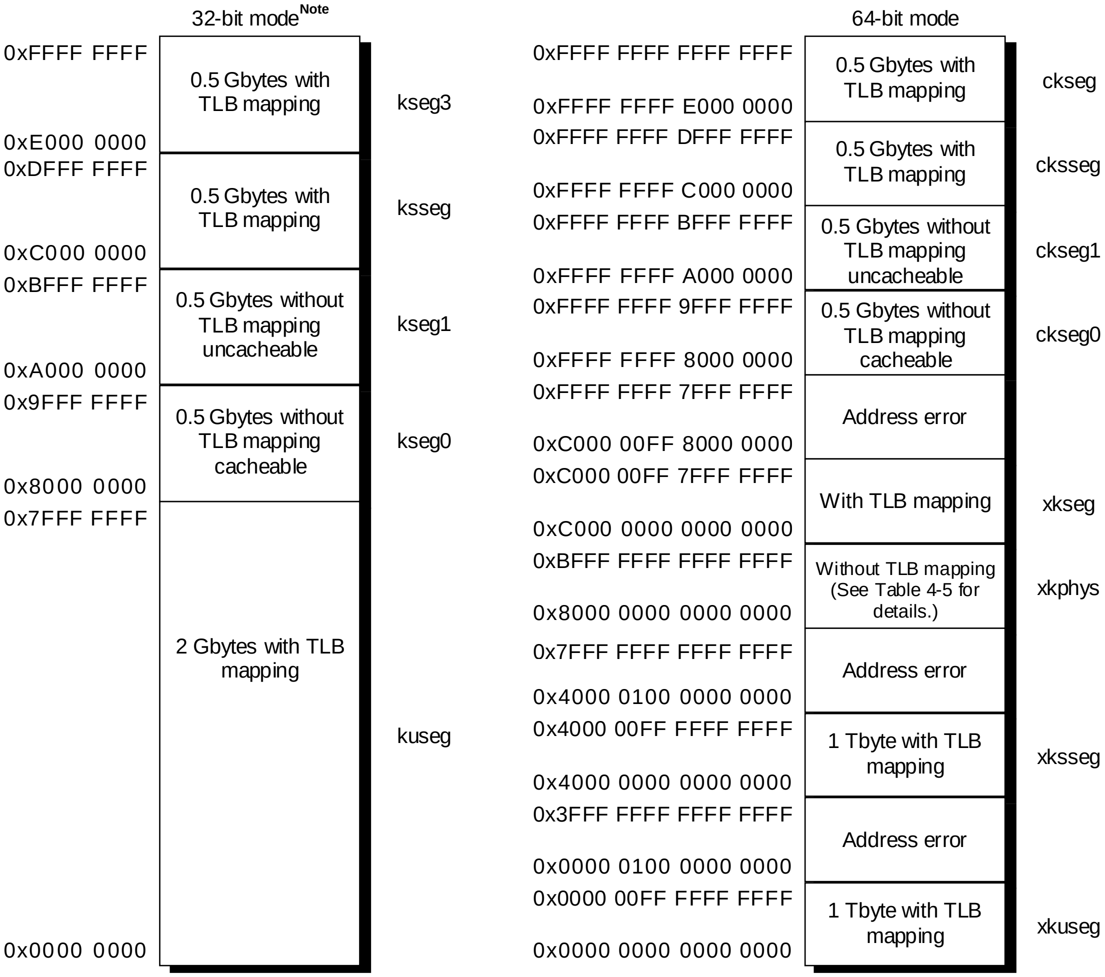
The addressing width in Kernel mode varies according to the state of the KX bit of the Status register, as follows:

- ✧ When KX = 0, 32-bit kernel space is selected.
- ✧ When KX = 1, 64-bit kernel space is selected.

The processor enters Kernel mode whenever an exception is detected and it remains in Kernel mode until an exception return (ERET) instruction is executed and results in ERL and/or EXL = 0. The ERET instruction restores the processor to the mode existing prior to the exception.

Kernel mode virtual address space is divided into regions differentiated by the high-order bits of the virtual address, as shown in Figure 4-7. Table 4-3 lists the characteristics of the 32-bit Kernel mode segments, and Table 4-4 lists the characteristics of the 64-bit Kernel mode segments.

Figure 4-7. Kernel Mode Address Space



Note The VR4101 uses 64-bit addresses within it. For 32-bit mode addressing, bit 31 is sign-extended to bits 32 to 63, and the resulting 32 bits are used for addressing. Usually, it is impossible for 32-bit mode programs to generate invalid addresses. In an operation of base register + offset for addressing, however, a two's complement overflow may occur, causing an invalid address. Note that the result becomes undefined. Two factors that can cause a two's complement follow:

- ✧ When offset bit 15 is 0, base register bit 31 is 0, and bit 31 of the operation “base register + offset” is 1
- ✧ When offset bit 15 is 1, base register bit 31 is 1, and bit 31 of the operation “base register + offset” is 0

Table 4-3. 32-bit Kernel Mode Segments

Address bit value	Status register bit value				Segment name	Virtual address	Physical address	Size
	KSU	EXL	ERL	KX				
32-bit A[31] = 0	KSU = 00 or EXL = 1 or ERL = 1			0	kuseg	0x0000 0000 to 0x7FFF FFFF	TLB map	2 Gbytes (2^{31} bytes)
32-bit A[31..29] = 100				0	kseg0	0x8000 0000 to 0x9FFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
32-bit A[31..29] = 101				0	kseg1	0xA000 0000 to 0xBFFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
32-bit A[31..29] = 110				0	ksseg	0xC000 0000 to 0xDFFF FFFF	TLB map	512 Mbytes (2^{29} bytes)
32-bit A[31..29] = 111				0	kseg3	0xE000 0000 to 0xFFFF FFFF	TLB map	512 Mbytes (2^{29} bytes)

(a) kuseg (32-bit Kernel mode, user space)

When KX = 0 in the Status register, and the most-significant bit of the virtual address space is 0, the kuseg virtual address space is selected; it is the current 2-Gbyte (2^{31}-byte) user address space.

The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

If the ERL bit of the Status register is 1, the user address space is assigned 2 Gbytes (2^{31} bytes) and becomes unmapped (with virtual addresses being used as physical addresses) and uncached so that the cache error handler can use it. This allows the Cache Error exception code to operate uncached using r0 as a base register.

(b) kseg0 (32-bit Kernel mode, kernel space 0)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 100, the kseg0 virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) physical space.

References to kseg0 are not mapped through the TLB; the physical address selected is defined by subtracting 0x8000 0000 from the virtual address.

The K0 field of the Config register controls cacheability.

(c) kseg1 (32-bit Kernel mode, kernel space 1)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 101, the kseg1 virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) physical space.

References to kseg1 are not mapped through the TLB; the physical address selected is defined by subtracting 0xA000 0000 from the virtual address.

Caches are disabled for accesses to these addresses, and main memory (or memory-mapped I/O device registers) are accessed directly.

(d) ksseg (32-bit Kernel mode, supervisor space)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 110, the ksseg virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

(e) kseg3 (32-bit Kernel mode, kernel space 3)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 111, the kseg3 virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) kernel virtual space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

Table 4-4. 64-bit Kernel Mode Segments

Address bit value	Status register bit value				Segmen t name	Virtual address	Physical address	Size
	KSU	EXL	ERL	KX				
64-bit A[63..62] = 00	KSU = 00 or EXL = 1 or ERL = 1			1	xksuseg	0x0000 0000 0000 0000 to 0x0000 00FF FFFF FFFF	TLB map	1 Tbyte (2^{40} bytes)
64-bit A[63..62] = 01				1	xksseg	0x4000 0000 0000 0000 to 0x4000 00FF FFFF FFFF	TLB map	1 Tbyte (2^{40} bytes)
64-bit A[63..62] = 10				1	xkphys	0x8000 0000 0000 0000 to 0xBFFF FFFF FFFF FFFF	0x0000 0000 to 0xFFFF FFFF	4 Gbytes (2^{32} bytes)
64-bit A[63..62] = 11				1	xkseg	0xC000 0000 0000 0000 to 0xC000 00FF 7FFF FFFF	TLB map	$2^{40} - 2^{31}$ bytes

64-bit A[63..62] = 11 A[63..31] = -1		1	ckseg0	0xFFFF FFFF 8000 0000 to 0xFFFF FFFF 9FFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
64-bit A[63..62] = 11 A[63..31] = -1		1	ckseg1	0xFFFF FFFF A000 0000 to 0xFFFF FFFF BFFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
64-bit A[63..62] = 11 A[63..31] = -1		1	cksseg	0xFFFF FFFF C000 0000 to 0xFFFF FFFF DFFF FFFF	TLB map	512 Mbytes (2^{29} bytes)
64-bit A[63..62] = 11 A[63..31] = -1		1	ckseg3	0xFFFF FFFF E000 0000 to 0xFFFF FFFF FFFF FFFF	TLB map	512 Mbytes (2^{29} bytes)

(f) xkuseg (64-bit Kernel mode, user space)

When KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 00, the xkuseg virtual address space is selected; it is the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

If the ERL bit of the Status register is 1, the user address space is assigned 2 Gbytes (2^{31} bytes) and becomes unmapped (with virtual addresses being used as physical addresses) and uncached so that the cache error handler can use it. This allows the Cache Error exception code to operate uncached using r0 as a base register.

(g) xksseg (64-bit Kernel mode, current supervisor space)

When KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 01, the xksseg address space is selected; it is the current supervisor address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

(h) xkphys (64-bit Kernel mode, physical spaces)

When the KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 10, the virtual address space is called xkphys and selected as either cached or uncached. If any of bits 58 to 32 of the address is 1, an attempt to access that address results in an address error.

Table 4-5. Cacheability and the xkphys Address Space

Bits 61-59	Cacheability	Start address
0	Cached	0x8000 0000 0000 0000
1	Cached	0x8800 0000 0000 0000
2	Uncached	0x9000 0000 0000 0000
3	Cached	0x9800 0000 0000 0000
4	Cached	0xA000 0000 0000 0000
5	Cached	0xA800 0000 0000 0000
6	Cached	0xB000 0000 0000 0000
7	Cached	0xB800 0000 0000 0000

(i) xkseg (64-bit Kernel mode, physical spaces)

When the KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 11, the virtual address space is called xkseg and selected as either of the following:

λ kernel virtual space, xkseg, the current kernel virtual space; the virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address

λ one of the four 32-bit kernel compatibility spaces, as described in the next section.

(j) 64-bit Kernel mode compatible spaces (ckseg0, ckseg1, cksseg, and ckseg3)

If the conditions listed below are satisfied in Kernel mode, ckseg0, ckseg1, cksseg, or ckseg3 (each having 512 Mbytes) is selected as a compatible space according to the state of the bits 30 and 29 (two low-order bits) of the address.

- ✧ The KX bit of the Status register is 1.
- ✧ Bits 63 and 62 of the 64-bit virtual address are 11.
- ✧ Bits 61 to 31 of the virtual address is -1.

λ ckseg0

This space is an unmapped region, compatible with the 32-bit mode kseg0 space. The K0 field of the Config register controls cacheability and coherency.

λ ckseg1

This space is an unmapped and uncached region, compatible with the 32-bit mode kseg1 space.

λ cksseg

This space is the current supervisor virtual space, compatible with the 32-bit mode ksseg space.

λ ckseg3

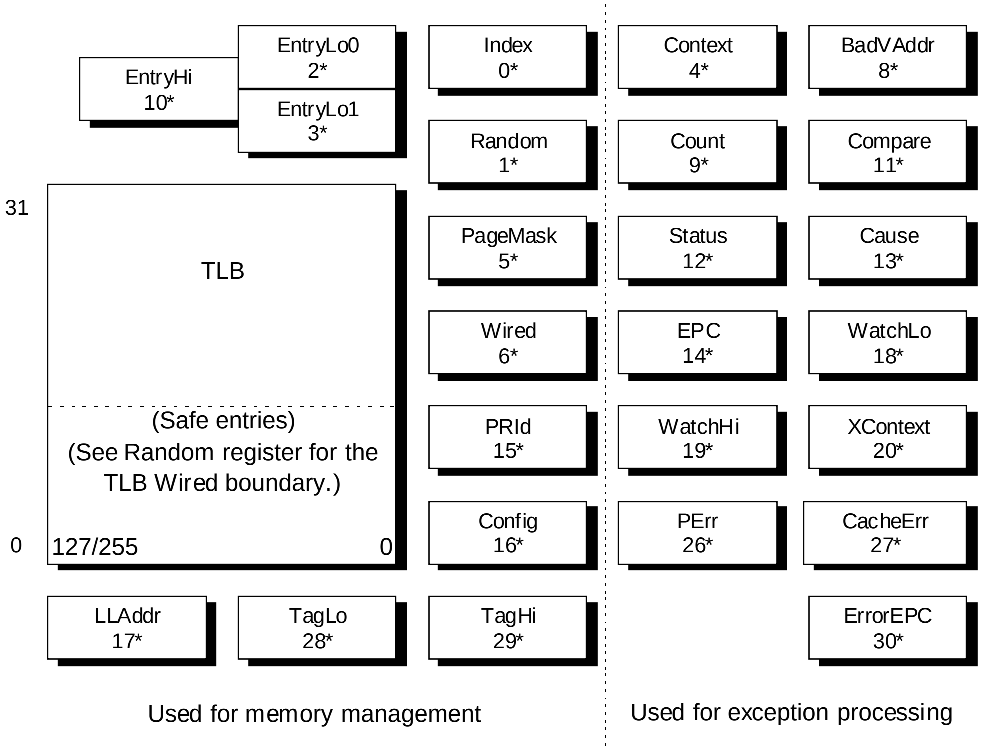
This space is the current supervisor virtual space, compatible with the 32-bit mode kseg3 space.

4.3 SYSTEM CONTROL COPROCESSOR

The System Control Coprocessor (CP0) is implemented as an integral part of the CPU, and supports memory management, address translation, exception handling, and other privileged operations. CP0 contains the registers shown in Figure 4-8 plus a 32-entry TLB. The sections that follow describe how the processor uses each of the memory management-related registers.

Remark Each CP0 register has a unique number that identifies it; this number is referred to as the register number. See Chapter 1 for details. Also see Chapter 5 for the CP0 functions and the relationships between exception processing and registers.

Figure 4-8. CP0 Registers and the TLB



Remark *: Register number

Caution For some instructions, pay attention to the interval between the instruction and the succeeding instruction when accessing the CP0 registers. This is because some time is required before the modification to the CP0 registers is reflected in the operation of the CPU. This is called the CP0 hazard. Refer to Chapter 25 for more details.

4.3.1 Format of a TLB Entry

Figure 4-9 shows the TLB entry formats for both 32- and 64-bit modes. Each field of an entry has a corresponding field in the EntryHi, EntryLo0, EntryLo1, or PageMask registers.

Figure 4-9. Format of a TLB Entry

(a) 32-bit mode

127 115	114 107	106 96
0	MASK	0
13	8	11

95 75	74	73 72	71 64
VPN2	G	0	ASID
21	1	2	8

63 60	59 38	37 35	34	33	32
0	PFN	C	D	V	0
4	22	3	1	1	1

31 28	27 6	5 3	2	1	0
0	PFN	C	D	V	0
4	22	3	1	1	1

(b) 64-bit mode

255 211	210 203	202 192
0	MASK	0
45	8	11

191 190	189 168	167 139	138	137 136	135 128
R	0	VPN2	G	0	ASID
2	22	29	1	2	8

127 92	91 70	69 67	66	65	64
0	PFN	C	D	V	0
36	22	3	1	1	1

63 28	27 6	5 3	2	1	0
0	PFN	C	D	V	0
36	22	3	1	1	1

The format of the EntryHi, EntryLo0, EntryLo1, and PageMask registers are nearly the same as the TLB entry. However, it is unknown what bit of the EntryHi register corresponds to the TLB G bit.

4.3.2 CP0 Registers

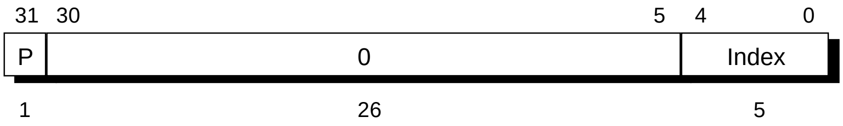
The CP0 registers explained below are accessed by the memory management system and software. A parenthesized number that follows each register name is a register number.

(1) Index register (0)

The Index register is a 32-bit, read/write register containing five bits to index an entry in the TLB. The most-significant bit of the register shows the success or failure of a TLB probe (TLBP) instruction.

The Index register also specifies the TLB entry affected by TLB read (TLBR) or TLB write index (TLBWI) instructions.

Figure 4-10. Index Register



P: Indicates whether probing is successful or not. It is set to 1 if the latest TLBP instruction fails. It is cleared to 0 when the TLBP instruction is successful.

Index: Specifies an index to a TLB entry that is a target of the TLBR or TLBWI instruction.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

(2) Random register (1)

The Random register is a read-only register. The low-order 5 bits are used in referencing a TLB entry. This register is decremented each time an instruction is executed. The values that can be set in the register are as follows:

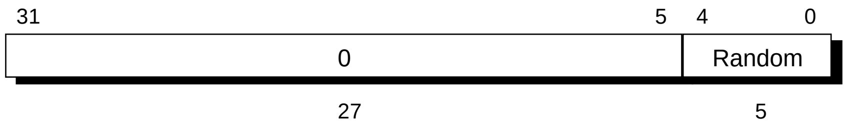
✧ The lower bound is the content of the Wired register.

✧ The upper bound is 31.

The Random register specifies the entry in the TLB that is affected by the TLBWR instruction. The register is readable to verify proper operation of the processor.

The Random register is set to the value of the upper bound upon Cold Reset. This register is also set to the upper bound when the Wired register is written. Figure 4-11 shows the format of the Random register.

Figure 4-11. Random Register



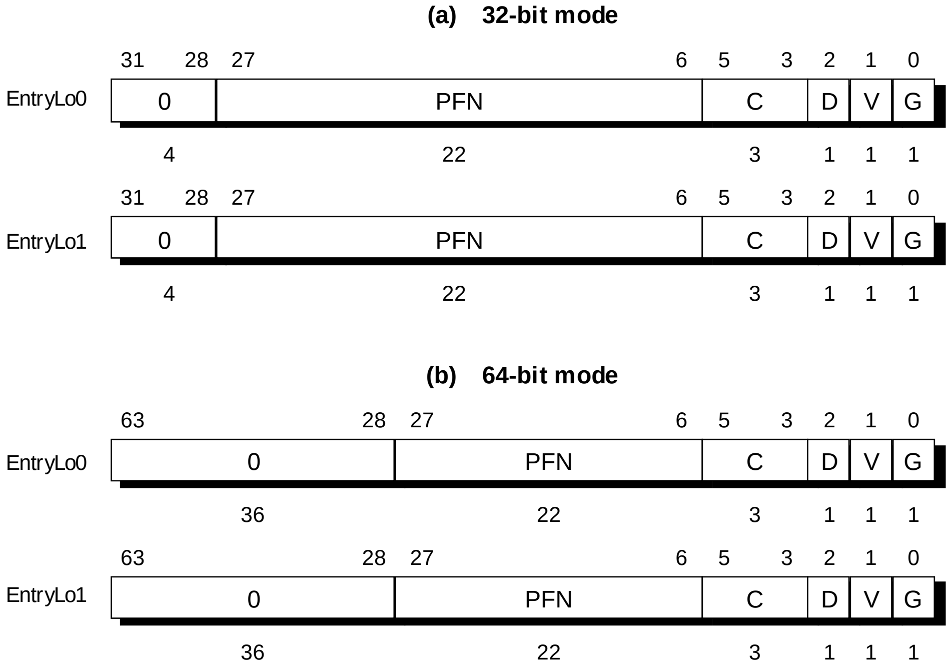
Random: TLB random index

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

(3) EntryLo0 (2) and EntryLo1 (3) registers

The EntryLo register consists of two registers that have identical formats: EntryLo0, used for even virtual pages and EntryLo1, used for odd virtual pages. The EntryLo0 and EntryLo1 registers are both read-/write-accessible. They are used to access the built-in TLB. When a TLB read/write operation is carried out, the EntryLo0 and EntryLo1 registers hold the contents of the low-order 32 bits of TLB entries at even and odd addresses, respectively.

Figure 4-12. EntryLo0 and EntryLo1 Registers



(a) 32-bit mode

(b) 64-bit mode

PFN: Page frame number; high-order bits of the physical address.

C: Specifies the TLB page attribute.

D: Dirty. If this bit is set to 1, the page is marked as dirty and, therefore, writable. This bit is actually a write-protect bit that software can use to prevent alteration of data.

V: Valid. If this bit is set to 1, it indicates that the TLB entry is valid; otherwise, a TLB Invalid exception (TLBL or TLBS) occurs.

G: Global. If this bit is set in both EntryLo0 and EntryLo1, then the processor ignores the ASID during TLB lookup.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

The coherency attribute (C) bits are used to specify whether to use the cache in referencing a page. When the cache is used, whether the page attribute is “cache used” or “cache not used” is selected by algorithm.

Table 4-6 lists the page attributes selected according to the value in the C bits.

Table 4-6. Cache Algorithm

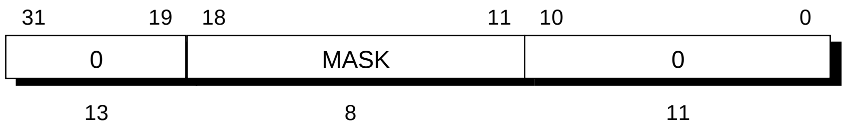
C bit value	Cache algorithm
0	Cache used
1	Cache used
2	Cache unusable
3	Cache used
4	Cache used
5	Cache used
6	Cache used
7	Cache used

(4) PageMask register (5)

The PageMask register is a read/write register used for reading from or writing to the TLB; it holds a comparison mask that sets the page size for each TLB entry, as shown in Table 4-7. Page sizes must be from 1 Kbyte to 256 Kbytes.

TLB read and write instructions use this register as either a source or a destination; Bits 18 to 11 that are targets of comparison are masked during address translation.

Figure 4-13. Page Mask Register



MASK: Page comparison mask, which determines the virtual page size for the corresponding entry.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

Table 4-7 lists the mask pattern for each page size. If the mask pattern is one not listed below, the TLB behaves unexpectedly.

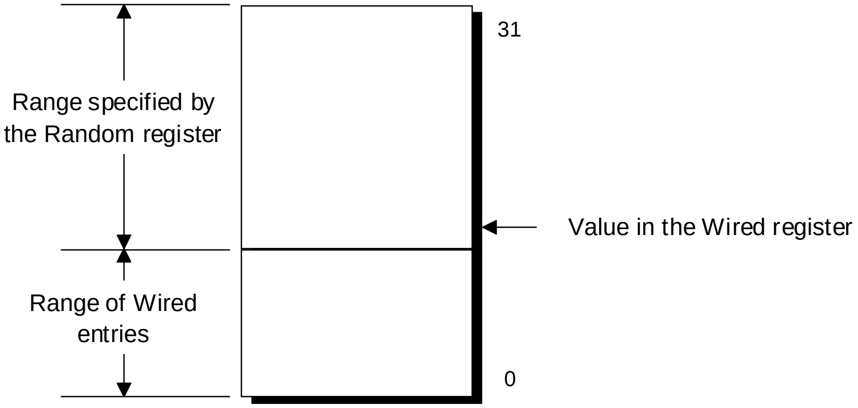
Table 4-7. Mask Values and Page Sizes

Page size	Bit							
	18	17	16	15	14	13	12	11
1 Kbyte	0	0	0	0	0	0	0	0
4 Kbytes	0	0	0	0	0	0	1	1
16 Kbytes	0	0	0	0	1	1	1	1
64 Kbytes	0	0	1	1	1	1	1	1
256 Kbytes	1	1	1	1	1	1	1	1

(5) Wired register (6)

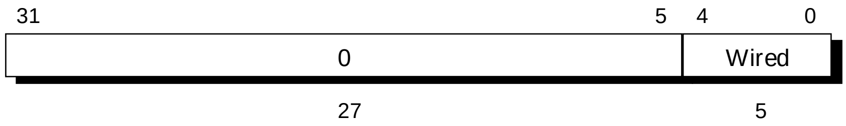
The Wired register is a read/write register that specifies the lower boundary of the random entry of the TLB as shown in Figure 4-14. Wired entries cannot be overwritten by a TLBWR instruction. They can, however, be overwritten by a TLBWI instruction. Random entries can be overwritten by both instructions.

Figure 4-14. Positions Indicated by the Wired Register


31
Range specified by the Random register
Value in the Wired register
Range of Wired entries
0

The Wired register is set to 0 upon Cold Reset. Writing this register also sets the Random register to the value of its upper bound (see Random register (1)). Figure 4-15 shows the format of the Wired register.

Figure 4-15. Wired Register



Wired: TLB wired boundary

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

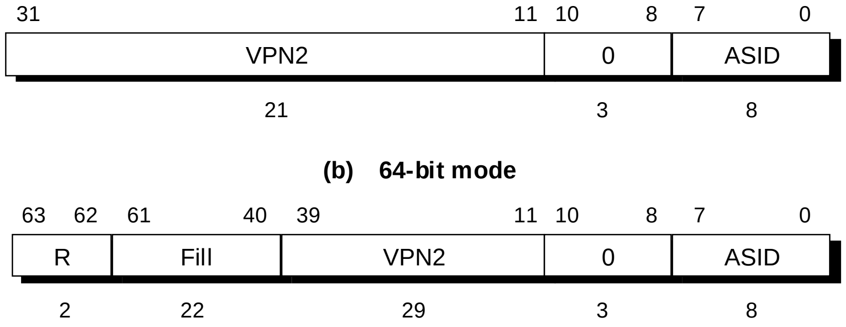
(6) EntryHi register (10)

The EntryHi register is write-accessible. It is used to access the built-in TLB. The EntryHi register holds the high-order bits of a TLB entry for TLB read and write operations. If a TLB Mismatch, TLB Invalid, or TLB Modified exception occurs, the EntryHi register holds the high-order bit of the TLB entry. The EntryHi register is also set with the virtual page number (VPN2) for a virtual address where an exception occurred and the ASID. See Chapter 5 for details of the TLB exception.

The ASID is used to read from or write to the ASID field of the TLB entry. It is also checked with the ASID of the TLB entry as the ASID of the virtual address during address translation.

The EntryHi register is accessed by the TLBP, TLBWR, TLBWI, and TLBR instructions.

Figure 4-16. EntryHi Register



(a) 32-bit mode

31 11	10 8	7 0
VPN2	0	ASID
21	3	8

(b) 64-bit mode

63 62	61 40	39 11	10 8	7 0
R	Fill	VPN2	0	ASID
2	22	29	3	8

VPN2: Virtual page number divided by two (mapping to two pages)

ASID: Address space ID. An 8-bit ASID field that lets multiple processes share the TLB; each process has a distinct mapping of otherwise identical virtual page numbers.

R: Space type (00 → user, 01 → supervisor, 11 → kernel). Matches bits 63 and 62 of the virtual address.

Fill: Reserved. Ignored on write. When read, returns zero.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

(7) Processor Revision Identifier (PRId) register (15)

The 32-bit, read-only Processor Revision Identifier (PRId) register contains information identifying the implementation and revision level of the CPU and CP0. Figure 4-17 shows the format of the PRId register.

Figure 4-17. PRId Register

31 16	15 8	7 0
0	Imp	Rev
16	8	8

Imp: CPU core processor ID number (0x0C for the VR4101)

Rev: CPU core processor revision number

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

The low-order byte (bits 7:0) of the PRId register is interpreted as a revision number, and the high-order byte (bits 15:8) is interpreted as an implementation number. The processor revision number is stored as a value in the form y.x, where y is a major revision number in bits 7 to 4 and x is a minor revision number in bits 3 to 0.

The revision number can distinguish some CPU core revisions, however there is no guarantee that changes to the CPU core will necessarily be reflected in the PRId register, or that changes to the revision number necessarily reflect real CPU core changes. Therefore, create a program that does not depend on the processor revision number area.

(8) Config register (16)

The Config register specifies various configuration options selected on VR4101 processors.

Some configuration options, as defined by the EC and BE fields, are set by the hardware during Cold Reset and are included in the Config register as read-only status bits for the software to access. Other configuration options are read/write (AD, EP, and K0 fields) and controlled by software; on Cold Reset these fields are undefined. Since only a subset of the VR4000 options are available in the VR4101, some bits are set to constants (e.g., bits 14:13) that were variable in the VR4000. The Config register should be initialized by software before caches are used. Figure 4-18 shows the format of the Config register.

Figure 4-18. Config Register Format

31	30 28	27 24	23	22 18	17	16	15	14	13 3	2 0
0	EC	EP	AD	0	1	0	BE	1	0	K0
1	3	4	1	5	1	1	1	1	11	3

EC: System clock ratio

0 → Processor clock frequency divided by 2

Others → Reserved

EP: Transfer data pattern (cache write-back pattern)

3 → DxDxDxDx

Others → Reserved

AD: Accelerate data mode

0 → VR4000 Series compatible mode

1 → Reserved

BE: BigEndianMem. Endian mode of memory and a kernel.

0 → Little endian

1 → Reserved

K0: kseg0 cache coherency algorithm

010 → Cache cannot be used.

Others → Cached

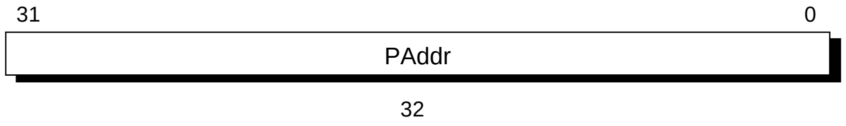
Note Be sure to set the EP and AD bits with 3 and 0, respectively. If they are set with any other values, the processor may behave unexpectedly.

(9) Load Linked Address (LLAddr) register (17)

The read/write Load Linked Address (LLAddr) register is not used with the VR4101 processor except for diagnostic purpose, and serves no function during normal operation.

LLAddr register is implemented just for a compatibility between the VR4101 and VR4000/VR4400.

Figure 4-19. LLAddr Register



PAddr: 32-bit physical address

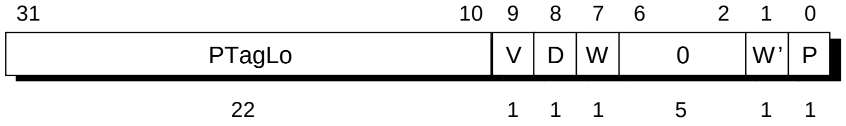
(10) Cache Tag registers (TagLo (28) and TagHi (29))

The TagLo and TagHi registers are 32-bit read/write registers that hold the primary cache tag and parity during cache initialization, cache diagnostics, or cache error processing. The Tag registers are written by the CACHE and MTC0 instructions.

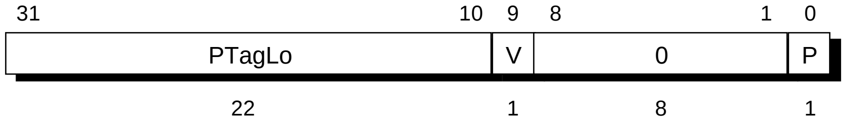
The P fields of these registers are ignored on Index Store Tag operations by the CACHE instruction. Parity is computed by the store operation. Figures 4-20 and 4-21 show the format of these registers.

Figure 4-20. TagLo Register

(a) When used with data cache



(b) When used with instruction cache



PTagLo: Specifies physical address bits 31 to 10.

V: Valid bit

D: Dirty bit. However, this bit is defined only for the compatibility with the VR4000 Series processors, and does not indicate the status of cache memory in spite of its readability and writability. This bit cannot change the status of cache memory.

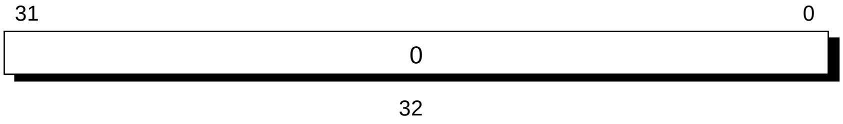
W: Write-back bit (set if cache line has been updated)

W': Odd parity for the write-back bit

P: Odd parity bit for primary cache tag

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

Figure 4-21. TagHi Register



0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

4.3.3 Virtual-to-Physical Address Translation

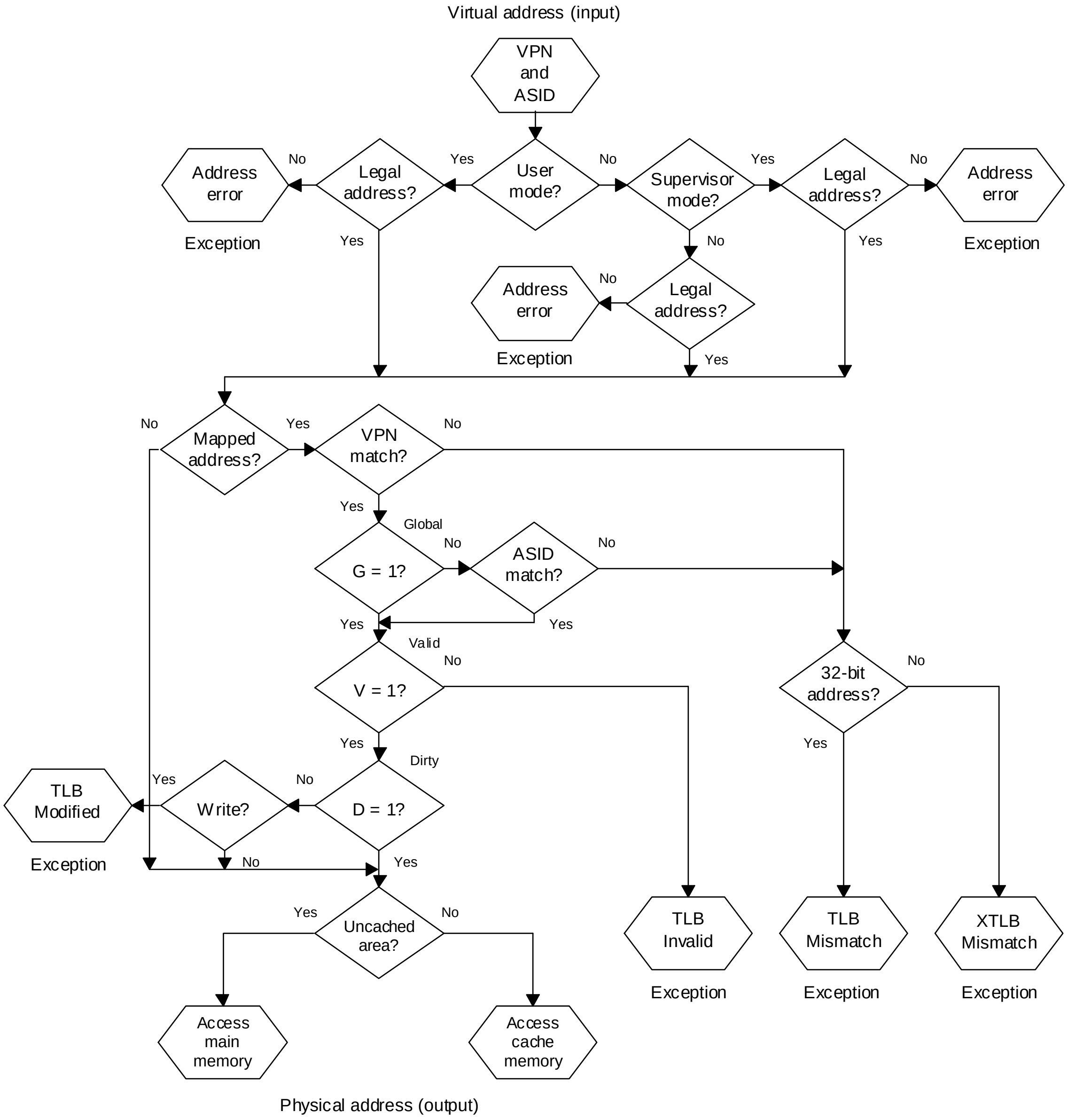
During virtual-to-physical address translation, the CPU compares the 8-bit ASID (if the Global bit, G, is not set to 1) of the virtual address to the ASID of the TLB entry to see if there is a match. One of the following comparisons are also made:

- In 32-bit mode, the high-order bits (up to bit 28, the number of bits depending upon the TLB page size) of the 32-bit virtual address are compared to the contents of the VPN2 (virtual page number divided by two) of each TLB entry.
- In 64-bit mode, the high-order (up to bit 39, the number of bits depending upon the TLB page size) of the 64-bit virtual address are compared to the contents of the R and the VPN2 (virtual page number divided by two) of each TLB entry.

If a TLB entry matches, the physical address and access control bits (C, D, and V) are retrieved from the matching TLB entry. While the V bit of the entry must be set to 1 for a valid address translation to take place, it is not involved in the determination of a matching TLB entry.

Figure 4-22 illustrates the TLB address translation flow.

Figure 4-22. TLB Address Translation



(1) TLB misses

If there is no TLB entry that matches the virtual address, a TLB Refill (miss) exception occurs[Note]. If the access control bits (D and V) indicate that the access is not valid, a TLB Modified or TLB Invalid exception occurs. If the C bit is 010, the retrieved physical address directly accesses main memory, bypassing the cache.

Note See Chapter 5 for details of the TLB Miss exception.

(2) TLB instructions

The instructions used for TLB control are described below.

(a) Translation lookaside buffer probe (TLBP)

The translation lookaside buffer probe (TLBP) instruction loads the Index register with a TLB number that matches the content of the EntryHi register. If there is no TLB number that matches the TLB entry, the highest-order bit of the Index register is set.

(b) Translation lookaside buffer read (TLBR)

The translation lookaside buffer read (TLBR) instruction loads the EntryHi, EntryLo0, EntryLo1, and PageMask registers with the content of the TLB entry indicated by the content of the Index register.

(c) Translation lookaside buffer write index (TLBWI)

The translation lookaside buffer write index (TLBWI) instruction writes the contents of the EntryHi, EntryLo0, EntryLo1, and PageMask registers to the TLB entry indicated by the content of the Index register.

(d) Translation lookaside buffer write random (TLBWR)

The translation lookaside buffer write random (TLBWR) instruction writes the contents of the EntryHi, EntryLo0, EntryLo1, and PageMask registers to the TLB entry indicated by the content of the Random register.

[MEMO]

CHAPTER 5 EXCEPTION PROCESSING

This chapter describes CPU exception processing, including an explanation of exception processing, followed by the format and use of each CPU exception register.

The chapter concludes with a description of each exception's cause, together with the manner in which the CPU processes and services each exception.

5.1 HOW EXCEPTION PROCESSING WORKS

The processor receives exceptions from a number of sources, including translation lookaside buffer (TLB) misses, arithmetic overflows, I/O interrupts, and system calls. When the CPU detects an exception, the normal sequence of instruction execution is suspended and the processor enters Kernel mode (see Chapter 4 for a description of system operating modes).

The processor then disables interrupts and transfers control for execution to the exception handler (located at a specific address as an exception handling routine implemented by software). The handler saves the context of the processor, including the contents of the program counter, the current operating mode (User or Supervisor), statuses, and interrupt enabling. This context is saved so it can be restored when the exception has been serviced.

When an exception occurs, the CPU loads the Exception Program Counter (EPC) register with a location where execution can restart after the exception has been serviced. The restart location in the EPC register is the address of the instruction that caused the exception or, if the instruction was executing in a branch delay slot, the address of the branch instruction immediately preceding the delay slot.

The VR4101 processor supports a Supervisor mode and fast TLB refill for all address spaces. The VR4101 also provides the following functions:

- ✧ Interrupt enable (IE) bit
- ✧ Operating mode (User, Supervisor, or Kernel)
- ✧ Exception level (normal or exception, as indicated by the EXL bit in the Status register)
- ✧ Error level (normal or error indicated by the ERL bit in the Status register).

Interrupts are enabled when the following conditions are satisfied:

- ✧ Interrupt enable bit (IE) = 1
- ✧ EXL bit = 0, ERL bit = 0
- ✧ Corresponding IM field bits in the Status register = 1
- ✧ The operating mode is specified by base mode when the exception level is normal (0), and is set to Kernel mode when either the EXL bit or ERL bit is set to 1.

Returning from an exception resets the exception level to normal.

The registers described later in the chapter assist in this exception processing by retaining address, cause and status information.

For a description of the exception handling process, see the description of the individual exception contained in this chapter, or the flowcharts at the end of this chapter.

5.2 PRECISION OF EXCEPTIONS

VR4101 exceptions are logically precise; the instruction that causes an exception and all those that follow it are aborted and can be re-executed after servicing the exception. When succeeding instructions are killed, exceptions associated with those instructions are also killed. Exceptions are not taken in the order detected, but in instruction fetch order.

There is a special case in which the VR4101 processor may not be able to restart easily after servicing an exception. When a cache data parity error exception occurs on a load with a cache hit, the VR4101 processor does not prevent the cache data (with erroneous parity) from being written back into the register file during the WB stage. The exception is still precise, since both the EPC and CacheError registers are updated with the correct virtual address pointing to the offending load instruction, and the exception handler can still determine the cause of exception and its origin. The program can be restarted by rewriting the destination register - not automatically, however, as in the case of all the other precise exceptions where no status change occurs.

5.3 EXCEPTION PROCESSING REGISTERS

This section describes the CP0 registers that are used in exception processing. Table 5-1 lists these registers, along with their number-each register has a unique identification number that is referred to as its register number. The CP0 registers not listed in the table are used in memory management (see Chapter 4 for details).

The exception handler examines the CP0 registers during exception processing to determine the cause of the exception and the state of the CPU at the time the exception occurred.

The registers in Table 5-1 are used in exception processing, and are described in the sections that follow.

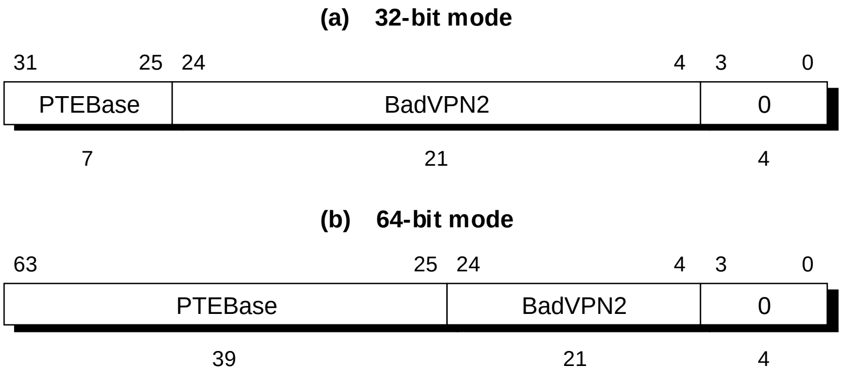
Table 5-1. CP0 Exception Processing Registers

Register name	Register number
Context register	4
BadVAddr register	8
Count register	9
Compare register	11
Status register	12
Cause register	13
EPC register	14
WatchLo register	18
WatchHi register	19
XContext register	20
Parity Error register	26
Cache Error register	27
ErrorEPC register	30

5.3.1 Context Register (4)

The Context register is a read/write register containing the pointer to an entry in the page table entry (PTE) array; this array is a table that stores virtual-to-physical address translations. When there is a TLB miss, the operating system loads the unsuccessfully translated entry from the PTE array to the TLB. The Context register is used by the TLB Refill exception handler for loading TLB entries. The Context register duplicates some of the information provided in the BadVAddr register, but the information is arranged in a form that is more useful for a software TLB exception handler. Figure 5-1 shows the format of the Context register.

Figure 5-1. Context Register Format



(a) 32-bit mode

(b) 64-bit mode

PTEBase: The PTEBase field is a read/write field. It is used by software as the pointer to the base address of the PTE table in the current user address space.

BadVPN2: The BadVPN2 field is written by hardware if a TLB miss occurs. This field holds the value (VPN2) obtained by halving the virtual page number of the most recent virtual address for which translation failed.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

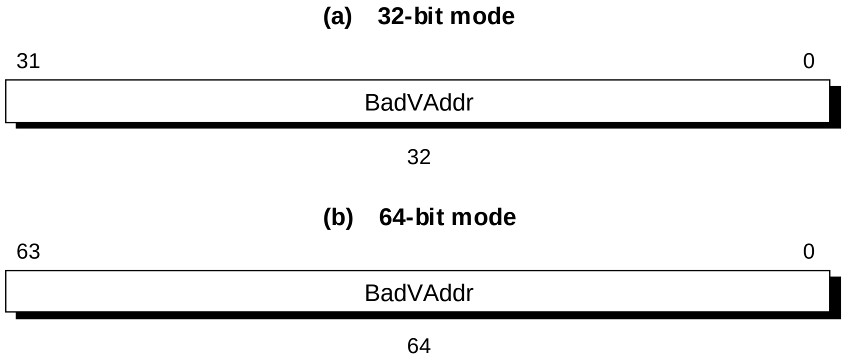
The 21-bit BadVPN2 field contains bits 31-11 of the virtual address that caused the TLB miss; bit 10 is excluded because a single TLB entry maps to an even-odd page pair. For a 1-Kbyte page size, this format can directly address the pair-table of 8-byte PTEs. When the page size is 4 Kbytes or more, shifting and masking this value produces the correct PTE reference address.

5.3.2 BadVAddr Register (8)

The Bad Virtual Address (BadVAddr) register is a read-only register that displays the most recent virtual address that failed to have a valid translation, or that had an addressing error. Figure 5-2 shows the format of the BadVAddr register.

Caution This register does not hold any information when a Bus Error exception occurs because it is not an Address Error exception.

Figure 5-2. BadVAddr Register Format



(a) 32-bit mode

(b) 64-bit mode

BadVAddr: Most recent virtual address for which an addressing error occurred, or for which address translation failed

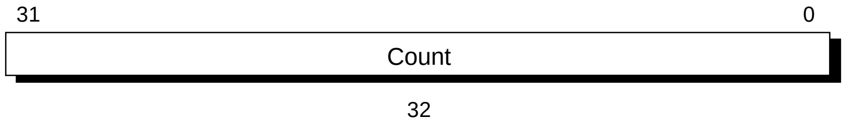
5.3.3 Count Register (9)

The read/write Count register acts as a timer. It is incremented at the MasterOut clock speed, regardless of whether instructions are being executed, retired, or any forward progress is actually made through the pipeline.

When the register reaches all ones, it rolls over to zero and continues counting. This register is used for self-testing, system initialization, or the establishment of inter-process synchronization.

Figure 5-3 shows the format of the Count register.

Figure 5-3. Count Register Format



Count: 32-bit counter value that is incremented in synchronization with the MasterOut clock

5.3.4 Compare Register (11)

The Compare register causes a timer interrupt; it maintains a stable value that does not change on its own.

When the value of the Count register (see Section 5.3.3) equals the value of the Compare register, the IP(7) bit in the Cause register is set. This causes an interrupt as soon as the interrupt is enabled.

Writing a value to the Compare register, as a side effect, clears the timer interrupt request.

For diagnostic purposes, the Compare register is a read/write register. Normally, this register is only used for a write. Figure 5-4 shows the format of the Compare register.

Figure 5-4. Compare Register Format

31 0
Compare
32

Compare: Value that is compared with the count value of the Count register

5.3.5 Status Register (12)

The Status register is a read/write register that contains the operating mode, interrupt enabling, and the diagnostic states of the processor. Figure 5-5 shows the format of the Status register. Figure 5-6 shows the details of the Diagnostic Status (DS) field. All DS field bits other than the TS bit are writeable.

Major Status register fields are detailed below.

Figure 5-5. Status Register Format

31 29	28	27 26	25	24 16	15 8	7	6	5	4 3	2	1	0
0	CU0	0	RE	DS	IM	KX	SX	UX	KSU	ERL	EXL	IE
3	1	2	1	9	8	1	1	1	2	1	1	1

CU0: Enables/disables the use of the coprocessor (1 → Enabled, 0 → Disabled).

CP0 can be used by the kernel at all times.

0: Reserved. To be set to 0.

RE: Enables/disables reversing of the endian setting in User mode (0 → Disabled, 1 → Enabled). This bit must be set to 0 since the VR4101 supports the little-endian order only.

DS: Diagnostic Status field (see Figure 5-6).

IM: Interrupt Mask field used to enable/disable interrupts (0 → Disabled, 1 → Enabled). This field consists of 8 bits that are used to control eight interrupts. The bits are assigned to interrupts as follows:

IM7 : Masks a timer interrupt.

IM(6:2) : Mask ordinary interrupts (Int(4:0)[Note] and write requests). However, Int(4:2)[Note] never occur in the VR4101.

IM(1:0) : Mask software interrupts.

Note: Int(4:0) are the internal signals of the VR4100 CPU core. For details about connection to the on-chip peripheral units, refer to Chapter 14.

KX: Enables 64-bit addressing in Kernel mode (0 → 32-bit, 1 → 64-bit). If this bit is set, an XTLB Refill exception occurs if a TLB miss occurs in the Kernel mode address space.

SX: Enables 64-bit addressing and operation in Supervisor mode (0 → 32-bit, 1 → 64-bit). If this bit is set, an XTLB Refill exception occurs if a TLB miss occurs in the Supervisor mode address space.

UX: Enables 64-bit addressing and operation in User mode (0 → 32-bit, 1 → 64-bit). If this bit is set, an XTLB Refill exception occurs if a TLB miss occurs in the User mode address space.

KSU: Sets and indicates the operating mode (10 → User, 01 → Supervisor, 00 → Kernel).

ERL: Sets and indicates the error level (0 → Normal, 1 → Error).

EXL: Sets and indicates the exception level (0 → Normal, 1 → Exception).

IE: Sets and indicates interrupt enabling/disabling (0 → Disabled, 1 → Enabled).

Figure 5-6. Status Register Diagnostic Status Field

24	23	22	21	20	19	18	17	16
0		BEV	TS	SR	0	CH	CE	DE
2		1	1	1	1	1	1	1

BEV: Specifies the base address of a TLB Refill exception vector and common exception vector (0 → Normal, 1 → Bootstrap).

TS: Causes the TLB to be shut down (read-only) (0 → Not shut down, 1 → Shut down). This bit is used to avoid any problems that may occur when multiple TLB entries match the same virtual address. After the TLB has been shut down, reset the processor to enable restart. Note that the TLB is shut down even if a TLB entry matching a virtual address is marked as being invalid (with the V bit cleared).

SR: Causes a Soft Reset or NMI exception (0 → Not caused, 1 → Caused).

CH: CP0 condition bit (0 → False, 1 → True). This bit can be read and written by software only; it cannot be accessed by hardware.

CE: When CE = 1, the contents of the PErr register are written to the check bits of the cache (See the description of the PErr register (26)).

DE: Specifies whether a cache parity error causes an exception (0 → Enable parity check, 1 → Disable parity check).

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

(1) Status register modes and access states

Fields of the Status register set the modes and access states described in the sections that follow.

(a) Interrupt enable

Interrupts are enabled when all of the following conditions are true:

- ✧ IE is set to 1.
- ✧ EXL is cleared to 0.
- ✧ ERL is cleared to 0.
- ✧ The appropriate bit of the IM is set to 1.

(b) Operating modes

The following Status register bit settings are required for User, Kernel, and Supervisor modes.

- ✧ The processor is in User mode when KSU = 10, EXL = 0, and ERL = 0.
- ✧ The processor is in Supervisor mode when KSU = 01, EXL = 0, and ERL = 0.
- ✧ The processor is in Kernel mode when KSU = 00, EXL = 1, or ERL = 1.

(c) 32- and 64-bit modes

The following Status register bit settings select 32- or 64-bit operation for User, Kernel, and Supervisor operating modes. Enabling 64-bit operation permits the execution of 64-bit opcodes and translation of 64-bit addresses. 64-bit operation for User, Kernel and Supervisor modes can be set independently.

- ✧ 64-bit addressing for Kernel mode is enabled when KX bit = 1. 64-bit operations are always valid in Kernel mode.
- ✧ 64-bit addressing and operations are enabled for Supervisor mode when SX bit = 1.
- ✧ 64-bit addressing and operations are enabled for User mode when UX bit = 1.

(d) Kernel address space accesses

Access to the kernel address space is allowed when the processor is in Kernel mode.

(e) Supervisor address space accesses

Access to the supervisor address space is allowed when the processor is in Supervisor or Kernel mode.

(f) User address space accesses

Access to the user address space is allowed in any of the three operating modes.

(2) Status register reset

The contents of the Status register are undefined after a Cold Reset, except for the following bits in the Diagnostic Status field:

- ✧ TS = 0, SR = 0
- ✧ ERL and BEV = 1
- ✧ The SR bit distinguishes between a Cold Reset and Soft Reset.

Remark Cold Reset and Soft Reset are the sequences to initialize the VR4100 CPU core. For details about initialization of the whole VR4101 including on-chip peripheral units, refer to Chapter 7.

5.3.6 Cause Register (13)

The 32-bit read/write Cause register describes the cause of the most recent exception. A 5-bit exception code indicates one of the causes (see Table 5-2). All bits in the Cause register, with the exception of the IP1 and IP0 bits, are read-only; IP1 and IP0 are used for software interrupts. Figure 5-7 shows the fields of this register; Table 5-2 describes the Cause register codes.

Figure 5-7. Cause Register Format

31	30	29 28	27 16	15 8	7	6 2	1 0
BD	0	CE	0	IP(IP7..IP0)	0	ExcCode	0
1	1	2	12	8	1	5	2

BD: Indicates whether the most recent exception occurred in the branch delay slot (1 → In delay slot, 0 → Normal).

CE: Indicates the number of the coprocessor for which a Coprocessor Unusable exception occurred. This field will remain undefined for as long as no exception occurs.

IP: Indicates whether an interrupt is pending (1 → Interrupt pending, 0 → No interrupt pending).

IP7 : A timer interrupt.

IP(6:2) : Ordinary interrupts (Int(4:0)[Note] and write requests). However, Int(4:2)[Note] never occur in the VR4101.

IP(1:0) : Software interrupts.

Note: Int(4:0) are the internal signals of the VR4100 CPU core. For details about connection to the on-chip peripheral units, refer to Chapter 14.

ExcCode: Exception code field

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

Table 5-2. Cause Register Exception Code Field

Exception code	Mnemonic	Description
0	Int	Interrupt exception
1	Mod	TLB Modified exception
2	TLBL	TLB Refill exception (load or fetch)
3	TLBS	TLB Refill exception (store)
4	AdEL	Address Error exception (load or fetch)
5	AdES	Address Error exception (store)
6	IBE	Bus Error exception (instruction fetch)
7	DBE	Bus Error exception (data load or store)
8	Sys	System Call exception
9	Bp	Breakpoint exception
10	RI	Reserved Instruction exception
11	CpU	Coprocessor Unusable exception
12	Ov	Integer Overflow exception
13	Tr	Trap exception
14-22	—	Reserved for future use
23	WATCH	Watch exception
24-31	—	Reserved for future use

5.3.7 Exception Program Counter (EPC) Register (14)

The Exception Program Counter (EPC) is a read/write register that contains the address at which processing resumes after an exception has been serviced.

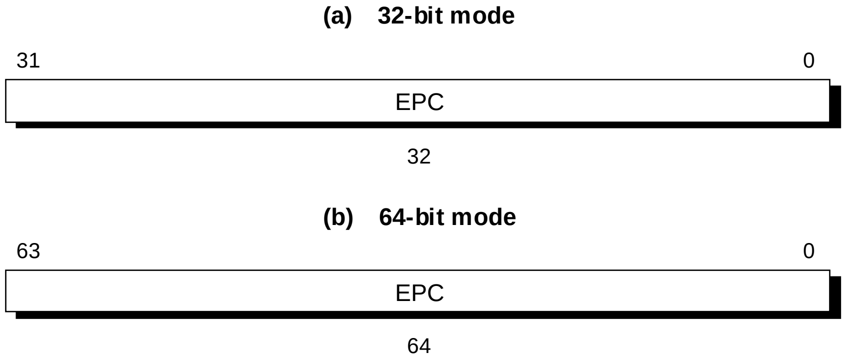
The EPC register contains either:

- Virtual address of the instruction that was the direct cause of the exception
- Virtual address of the immediately preceding branch or jump instruction (when the instruction associated with the exception is in a branch delay slot, and the BD bit in the Cause register is set to 1).

The EXL bit in the Status register is set to 1 to keep the processor from overwriting the address of the exception-causing instruction contained in the EPC register in the event of another exception.

Figure 5-8 shows the format of the EPC register.

Figure 5-8. EPC Register Format



(a) 32-bit mode

(b) 64-bit mode

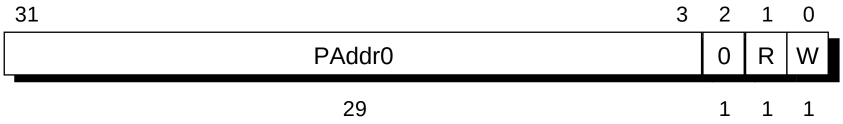
EPC: Restart address after exception processing

5.3.8 WatchLo (18) and WatchHi (19) Registers

The VR4101 processor provides a debugging feature to detect references to a selected physical address; load and store instructions to the location specified by the WatchLo and WatchHi registers cause a Watch exception.

Figures 5-9 and 5-10 show the format of the WatchLo and WatchHi registers.

Figure 5-9. WatchLo Register Format



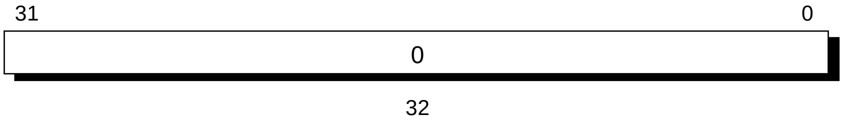
PAddr0: Specifies physical address bits 31 to 3.

R: If this bit is set to 1, an exception will occur when a load instruction is executed.

W: If this bit is set to 1, an exception will occur when a store instruction is executed.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

Figure 5-10. WatchHi Register Format



0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

5.3.9 XContext Register (20)

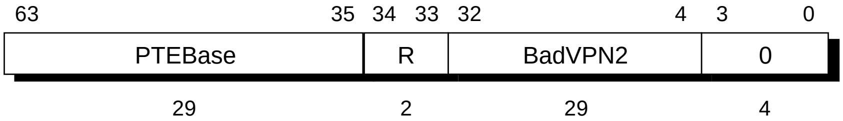
The read/write XContext register contains a pointer to an entry in the kernel page table entry (PTE) array, an operating system data structure that stores virtual-to-physical address translations. If a TLB miss occurs, the operating system loads the untranslated data from the PTE into the TLB to handle the software error.

The XContext register is used by the XTLB Refill exception handler to load TLB entries in 64-bit addressing mode.

The XContext register duplicates some of the information provided in the BadVAddr register, and puts it in a form useful for the XTLB exception handler.

This register is included solely for operating system use. The operating system sets the PTEBase field in the register, as needed. Figure 5-11 shows the format of the XContext register.

Figure 5-11. XContext Register Format



PTEBase: The PTEBase field is a read/write field, and is used by software as the pointer to the base address of the PTE table in the current user address space.

BadVPN2: The BadVPN2 field is written by hardware if a TLB miss occurs. This field holds the value (VPN2) obtained by halving the virtual page number of the most recent virtual address for which translation failed.

R: Space type (00 → User, 01→ Supervisor, 11 → Kernel). The setting of this field matches virtual address bits 63 and 62.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

The 29-bit BadVPN2 field has bits 39 to 11 of the virtual address that caused the TLB miss; bit 10 is excluded because a single TLB entry maps to an even-odd page pair. For a 1-Kbyte page size, this format may be used directly to address the pair-table of 8-byte PTEs. For 4-Kbyte-or-more page and PTE sizes, shifting and masking this value produces the appropriate address.

5.3.10 Parity Error Register (26)

The read/write Parity Error (PErr) register contains the cache data parity bits for cache initialization, cache diagnostics, or cache error processing.

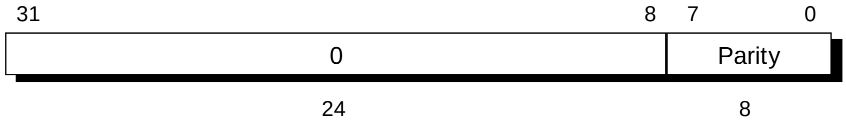
The PErr register is loaded by the Index_Load_Tag CACHE instruction. All bit of the parity field are valid on the data cache operation. But a LSB of the parity field is valid on the instruction cache operation.

The contents of the PErr register are:

- written into the primary data cache on store instructions (instead of the computed parity) when the CE bit of the Status register is set to 1
- substituted for the computed parity for the CACHE Fill instruction

Figure 5-12 shows the format of the PErr register.

Figure 5-12. PErr Register Format



Parity: Specifies the 8-bit parity data to be read from or written to the primary cache.

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

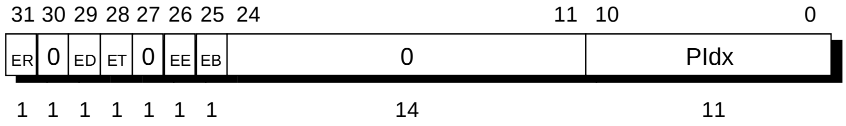
5.3.11 Cache Error Register (27)

The 32-bit read/write Cache Error (CacheErr) register processes parity errors in the primary cache. Parity errors cannot be corrected by on-chip hardware.

The CacheErr register holds cache index and status bits that indicate the cause of the error.

Figure 5-13 shows the format of the CacheErr register.

Figure 5-13. CacheErr Register Format



ER: Reference type (0 → Instruction, 1 → Data)

ED: Indicates whether an error occurred in the data field (0 → Normal, 1 → Error).

ET: Indicates whether an error occurred in the tag field (0 → Normal, 1 → Error).

EE: This bit is set if an error occurs on the SysAD bus.

EB: This bit is set if a data error occurs subsequent to an instruction error. (The error status is indicated by the remaining bit positions.) In this case, the data cache must be flushed upon the completion of instruction error processing.

PIdx: Cache index

0: Reserved for future use. Write 0 in a write operation. When this field is read, 0 is read.

5.3.12 ErrorEPC Register (30)

The Error Exception Program Counter (ErrorEPC) register is similar to the EPC register. It is also used to store the Cache error, Cold Reset, Soft Reset, and Program Counter on NMI exceptions.

The read/write ErrorEPC register contains the virtual address at which instruction processing can resume after servicing an error. This address can be:

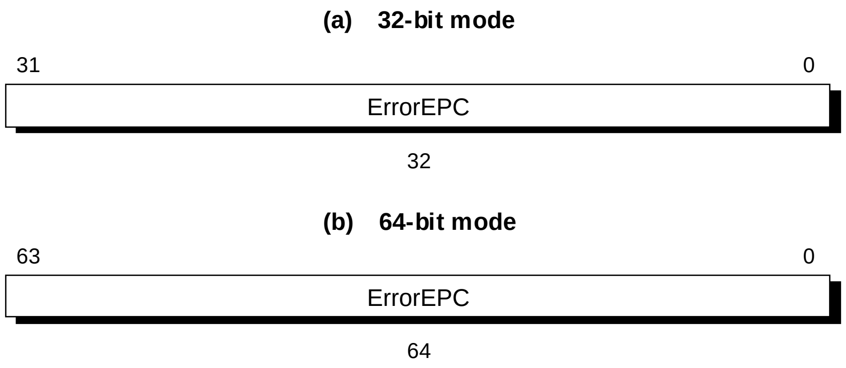
- the virtual address of the instruction that caused the exception
- the virtual address of the immediately preceding branch or jump instruction, when the instruction associated with the error exception is in a branch delay slot.

The contents of the ErrorEPC register do not change when the ERL bit of the Status register is set to 1. This prevents the processor when other exceptions occur from overwriting the address of the instruction in this register which causes an error exception.

There is no branch delay slot indication for the ErrorEPC register.

Figure 5-14 shows the format of the ErrorEPC register.

Figure 5-14. The ErrorEPC Register Format


(a) 32-bit mode

(b) 64-bit mode

ErrorEPC: Restart address after parity error exception processing, or the contents of the Program Counter at Cold Reset, Soft Reset, or NMI exception.

5.4 DETAILS OF EXCEPTIONS

This section gives sample exception handler operations for the following exception types:

5.4.1 Exception Types

This section gives sample exception handler operations for the following exception types:

- ✧ Cold Reset
- ✧ Soft Reset
- ✧ NMI
- ✧ Cache error
- ✧ Remaining processor exceptions

When the EXL and ERL bits in the Status register are 0, either User, Supervisor, or Kernel operating mode is specified by the KSU bits in the Status register. When either the EXL or ERL bit is set to 1, the processor is in Kernel mode.

When the processor takes an exception, the EXL bit is set to 1, meaning the system is in Kernel mode. After saving the appropriate state, the exception handler typically resets the EXL bit back to 0. The exception handler sets the EXL bit to 1 so that the saved state is not lost upon the occurrence of another exception while the saved state is being restored.

Returning from an exception also resets the EXL bit to 0. For details, see Chapter 24.

5.4.2 Exception Vector Locations

The Cold Reset, Soft Reset, and NMI exceptions are always vectored to the following reset exception vector address. This address is in an uncached, unmapped space.

- ✧ 0xBFC0 0000 in 32-bit mode
- ✧ 0xFFFF FFFF BFC0 0000 in 64-bit mode

Addresses for the remaining exceptions are a combination of a vector offset and a base address.

(1) TLB Refill vector

When BEV bit = 0, the vector base address for the TLB Refill exception is in kseg0 (unmapped) space.

✧ 0x8000 0000 in 32-bit mode
✧ 0xFFFF FFFF 8000 0000 in 64-bit mode

When BEV bit = 1, the vector base address for the TLB Refill exception is in kseg1 (uncached, unmapped) space.

✧ 0xBFC0 0200 in 32-bit mode
✧ 0xFFFF FFFF BFC0 0200 in 64-bit mode

This is an uncached, non-TLB-mapped space, allowing the exception handler to bypass the cache and TLB.

(2) Cache Error exception vector

When BEV bit = 0, the vector base address for the Cache Error exception is in kseg1 (uncached, unmapped) space.

✧ 0xA000 0000 in 32-bit mode
✧ 0xFFFF FFFF A000 0000 in 64-bit mode

When BEV bit = 1, the vector base address for the Cache Error exception is in kseg1 (uncached, unmapped) space.

✧ 0xBFC0 0200 in 32-bit mode
✧ 0xFFFF FFFF BFC0 0200 in 64-bit mode

This is an uncached, non-TLB-mapped space, allowing the exception handler to bypass the cache and TLB.

64-bit mode exception vectors and their offsets are shown below.

Table 5-3. 64-Bit Mode Exception Vector Base Addresses

	Vector base address	Vector offset
Cold Reset Soft Reset NMI	0xFFFF FFFF BFC0 0000	0x0000
Cache Error	0xFFFF FFFF A000 0000 (BEV=0) 0xFFFF FFFF BFC0 0200 (BEV=1)	0x0100
TLB Refill (EXL = 0)	0xFFFF FFFF 8000 0000 (BEV=0) 0xFFFF FFFF BFC0 0200 (BEV=1)	0x0000
XTLB Refill (EXL = 1)		0x0080
Other exceptions		0x0180

5.4.3 Priority of Exceptions

The remainder of this chapter describes exceptions in the order of their priority shown in Table 5-4 (certain of the exceptions, such as the TLB exceptions and Instruction/Data exceptions, grouped together for convenience). While more than one exception can occur for a single instruction, only the exception with the highest priority is reported. Table 5-4 lists the priorities.

Table 5-4. Exception Priority Order

High	Cold Reset
↑	Soft Reset
│	NMI
│	Address Error (instruction fetch)
│	TLB/XTLB Refill (instruction fetch)
│	TLB Invalid (instruction fetch)
│	Cache Error (instruction fetch)
│	Bus Error (instruction fetch)
│	System Call
│	Breakpoint
│	Coprocessor Unusable
│	Reserved Instruction
│	Trap
│	Integer Overflow
│	Address Error (data access)
│	TLB/XTLB Refill (data access)
│	TLB Invalid (data access)
│	TLB Modified (data write)
│	Cache Error (data access)
│	Watch
│	Bus Error (data access)
│	Interrupt (other than NMI)
Low	

Generally speaking, the exceptions described in the following sections are handled (“processed”) by hardware; these exceptions are serviced by software.

5.4.4 Cold Reset Exception

Cause

The Cold Reset exception occurs when the ColdReset* signal (internal) is asserted and then deasserted. This exception is not maskable. The Reset* signal (internal) must be asserted along with the ColdReset* signal (for details, see Chapter 7).

Processing

The CPU provides a special interrupt vector for this exception:

- 0xBFC0 0000 in 32-bit mode
- 0xFFFF FFFF BFC0 0000 in 64-bit mode

The Cold Reset vector resides in unmapped and uncached CPU address space, so the hardware need not initialize the TLB or the cache to process this exception. It also means the processor can fetch and execute instructions while the caches and virtual memory are in an undefined state.

The contents of all registers in the CPU are undefined when this exception occurs, except for the following register fields:

- TS and SR of the Status register are cleared to 0.
- ERL and BEV of the Status register are set to 1.
- The Random register is initialized to the value of its upper bound (31).
- The Wired register is initialized to 0.
- Bits 31 to 28 of the Config register are set to 0, and bits 22 to 3 to 0x04800.
- All other bits are undefined.

Servicing

The Cold Reset exception is serviced by:

- Initializing all processor registers, coprocessor registers, TLB, caches, and the memory system
- Performing diagnostic tests
- Bootstrapping the operating system

5.4.5 Soft Reset Exception

Cause

A Soft Reset (sometimes called Warm Reset) occurs when the ColdReset* signal remains deasserted while the Reset* signal goes from assertion for at least 16 MasterClocks to deassertion (for details, see Chapter 7).

A Soft Reset immediately resets all state machines, and sets the SR bit of the Status register. Execution begins at the reset vector when the reset is deasserted. This exception is not maskable.

Processing

The CPU provides a special interrupt vector for this exception (same location as Cold Reset):

- ✧ 0xBFC0 0000 in 32-bit mode
- ✧ 0xFFFF FFFF BFC0 0000 in 64-bit mode

This vector is located within unmapped and uncached address space, so that the cache and TLB need not be initialized to process this exception. The SR bit of the Status register is set to 1 to distinguish this exception from a Cold Reset exception.

When this exception occurs, the contents of all registers are preserved except for the following registers:

- ✧ In the ErrorEPC register, the value of the Program Counter when an exception occurred is set.
- ✧ TS bit of the Status register are cleared to 0.
- ✧ ERL, SR, and BEV bits of the Status register are set to 1.

A Soft Reset can occur when the processor is placed in any state. So, access to the operating cache or system interface may be aborted. This means that the contents of the cache and memory will be unpredictable if a Soft Reset occurs.

Servicing

The Soft Reset exception is serviced by:

- ✧ Preserving the current processor states for diagnostic tests
- ✧ Reinitializing the system in the same way as for a Cold Reset exception

5.4.6 NMI Exception

Cause

The Nonmaskable Interrupt (NMI) exception occurs in response to the input of the NMI signal (internal). An NMI can also be set by an external write through the SysAD bus. This interrupt is not maskable; it occurs regardless of the settings of the EXL, ERL, and the IE bits in the Status register (for details, see Chapters 9 and 14).

Processing

The CPU provides a special interrupt vector for this exception:

- ✧ 0xBFC0 0000 in 32-bit mode
- ✧ 0xFFFF FFFF BFC0 0000 in 64-bit mode

This vector is located within unmapped and uncached address space so that the cache and TLB need not be initialized to process an NMI interrupt. The SR bit of the Status register is set to 1 to distinguish this exception from a Cold Reset exception.

Unlike Cold Reset and Soft Reset, but like other exceptions, NMI is taken only at instruction boundaries. The state of the caches and memory system are preserved by this exception.

When this exception occurs, the contents of all registers are preserved except for the following registers:

- ✧ In the ErrorEPC register, the value of the Program Counter when an exception occurred is set.
- ✧ The TS bit of the Status register is cleared to 0.
- ✧ The ERL, SR, and BEV bits of the Status register are set to 1.

Servicing

The NMI exception is serviced by:

- ✧ Preserving the current processor states for diagnostic tests
- ✧ Reinitializing the system in the same way as for a Cold Reset exception

5.4.7 Address Error Exception

Cause

The Address Error exception occurs when an attempt is made to execute one of the following. This exception is not maskable.

- Execution of the LW, LWU, SW, or CACHE instruction for word data that is not located on a word boundary
- Execution of the LH, LHU, or SH instruction for half-word data that is not located on a half-word boundary
- Execution the LD or SD instruction for double-word data that is not located on a double-word boundary
- Referencing the kernel address space in User or Supervisor mode
- Referencing the supervisor space in User mode
- Referencing an address that does not exist in the kernel, user, or supervisor address space in 64-bit Kernel, User, or Supervisor mode
- Branching to an address that is not located on a word boundary

Processing

The common exception vector is used for this exception. The AdEL or AdES code in the Cause register is set, indicating whether the instruction caused the exception with an instruction reference (AdEL), load operation (AdEL), or store operation (AdES).

When this exception occurs, the BadVAddr register retains the virtual address that was not properly aligned or was referenced in protected address space. The contents of the VPN field of the Context and EntryHi registers are undefined, as are the contents of the EntryLo register.

The EPC register contains the address of the instruction that caused the exception, unless this instruction is in a branch delay slot. If it is in a branch delay slot, the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

The kernel reports the UNIX™ SIGSEGV (segmentation violation) signal to the current process, but the exception is usually fatal.

5.4.8 TLB Exceptions

Three types of TLB exceptions can occur:

- TLB Refill exception occurs when there is no TLB entry that matches a referenced address.
- A TLB Invalid exception occurs when a TLB entry that matches a referenced virtual address is marked as being invalid (with the V bit set to 0).
- The TLB Modified exception occurs when a TLB entry that matches a virtual address referenced by the store instruction is marked as being valid (with the V bit set to 1), but is not writeable.

The following three sections describe these TLB exceptions.

(1) TLB Refill exception (32-bit space mode)/XTLB Refill exception (64-bit space mode)

Cause

The TLB Refill exception occurs when there is no TLB entry to match a reference to a mapped address space. This exception is not maskable.

Processing

There are two special exception vectors for this exception; one for references to 32-bit address spaces, and one for references to 64-bit address spaces. The UX, SX, and KX bits of the Status register determine whether the user, supervisor or kernel address spaces referenced are 32-bit or 64-bit spaces. When the EXL bit of the Status is set to 0, these two special vectors are referenced. When the EXL bit is set to 1, the common exception vector is referenced.

This exception sets the TLBL or TLBS code in the ExcCode field of the Cause register. If this exception has been caused by an instruction reference or load operation, TLBL is set. If it has been caused by a store operation, TLBS is set.

When this exception occurs, the BadVAddr, Context, XContext and EntryHi registers hold the virtual address that failed address translation. The EntryHi register also contains the ASID from which the translation fault occurred. The Random register normally contains a valid location in which to place the replacement TLB entry. The contents of the EntryLo register are undefined.

The EPC register contains the address of the instruction that caused the exception, unless this instruction is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

To service this exception, the contents of the Context or XContext register are used as a virtual address to fetch memory words containing the physical page frame and access control bits for a pair of TLB entries. The memory word is written into the TLB entry by using the EntryLo0, EntryLo1, or EntryHi register.

It is possible that the physical page frame and access control bits are placed in a page where the virtual address is not resident in the TLB. This condition is processed by allowing a TLB Refill exception in the TLB Refill exception handler. In this case, the common exception vector is used because the EXL bit of the Status register is set to 1.

(2) TLB Invalid exception

Cause

The TLB Invalid exception occurs when a virtual address reference matches a TLB entry that is marked invalid (the V bit is set to 0). This exception is not maskable.

Processing

The common exception vector is used for this exception. The TLBL or TLBS code in the ExcCode field of the Cause register is set. If this exception has been caused by an instruction reference or load operation, TLBL is set. If it has been caused by a store operation, TLBS is set.

When this exception occurs, the BadVAddr, Context, XContext and EntryHi registers contain the virtual address that failed address translation. The EntryHi register also contains the ASID from which the translation fault occurred. The Random register normally contains a valid location in which to place the replacement TLB entry. The contents of the EntryLo register are undefined.

The EPC register contains the address of the instruction that caused the exception unless this instruction is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

Usually, the V bit of a TLB entry is cleared in the following cases:

- When a virtual address does not exist
- When the virtual address exists, but is not in main memory (a page fault)
- When a trap is desired on any reference to the page (for example, to maintain a reference bit)

After servicing the cause of a TLB Invalid exception, the TLB entry is located with a TLBP (TLB Probe) instruction, and replaced by an entry with its Valid bit set to 1.

(3) TLB Modified exception

Cause

The TLB Modified exception occurs when a virtual address referenced by the store instruction matches a TLB entry that is marked valid (the V bit is set to 1) but is not writeable (the D bit is set to 0). This exception is not maskable.

Processing

The common exception vector is used for this exception, and the Mod code in the ExcCode field of the Cause register is set.

When this exception occurs, the BadVAddr, Context, XContext and EntryHi registers contain the virtual address that failed address translation. The EntryHi register also contains the ASID from which the translation fault occurred. The contents of the EntryLo register are undefined.

The EPC register contains the address of the instruction that caused the exception unless that instruction is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

The kernel uses the failed virtual address or virtual page number to identify the corresponding access control bits. The page identified may or may not permit write accesses; if writes are not permitted, a write protection violation occurs.

If write accesses are permitted, the page frame is marked dirty (/writeable) by the kernel in its own data structures.

The TLBP instruction places the index of the TLB entry that must be altered into the Index register. The EntryLo register is loaded with a word containing the physical page frame and access control bits (with the D bit set to 1), and the EntryHi and EntryLo registers are written into the TLB.

5.4.9 Cache Error Exception

Cause

The Cache Error exception occurs when either a primary cache parity error or a System bus parity error is detected. This exception is not maskable, but error detection may be disabled by the DE bit of the Status register.

If a parity error is detected when the DE bit of Status register is not set, a cache error exception is taken during one of the following operations:

- ✧ An instruction fetch from instruction cache
- ✧ A load from the data cache
- ✧ Tag parity check on a store
- ✧ Main memory read by the processor
- ✧ Most of the CACHE instructions (no exception is taken for the Index_Load_Tag and Index_Store_Tag CACHE instructions)

Processing

The processor sets the ERL bit in the Status register, saves the address to recover from the exception in ErrorEPC register, and then transfers to a special vector in uncached space.

If the BEV bit = 0, the vector is one of the following:

- ✧ 0xA000 0100 in 32-bit mode
- ✧ 0xFFFF FFFF A000 0100 in 64-bit mode

If the BEV bit = 1, the vector is one of the following:

- ✧ 0xBFC0 0300 in 32-bit mode
- ✧ 0xFFFF FFFF BFC0 0300 in 64-bit mode

Servicing

All errors should be logged. To correct cache parity errors, the system uses the CACHE instruction to invalidate the cache block, overwrites the old data through a cache miss, and resumes execution with an ERET instruction. Other errors are not correctable and are likely to be fatal to the current process.

5.4.10 Bus Error Exception

Cause

A Bus Error exception is raised by board-level circuitry for events such as bus time-out, local bus parity errors, and invalid physical memory addresses or access types. This exception is not maskable.

A Bus Error exception occurs only when a cache miss refill, uncached reference, or unbuffered write occurs synchronously.

Processing

The common interrupt vector is used for a Bus Error exception. The IBE or DBE code in the ExcCode field of the Cause register is set, signifying whether the instruction caused the exception by an instruction reference, load operation, or store operation.

The EPC register contains the address of the instruction that caused the exception, unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

The physical address at which the fault occurred can be computed from information available in the System Control Coprocessor (CP0) registers.

- ✧ If the IBE code in the Cause register is set (indicating an instruction fetch), the virtual address is contained in the EPC register (or 4 + the contents of the EPC register if the BD bit of the Cause register is set to 1).
- ✧ If the DBE code is set (indicating a load or store), the virtual address of the instruction that caused the exception (the address of the preceding branch instruction if the BD bit of the Cause register is set to 1) is contained in the EPC register (or 4 + the contents of the EPC register if the BD bit of the Cause register is set to 1).

The virtual address of the load and store instruction can then be obtained by interpreting the instruction. The physical address can be obtained by using the TLBP instruction and reading the EntryLo register to compute the physical page number.

At the time of this exception, the kernel reports the UNIX SIGBUS (bus error) signal to the current process, but the exception is usually fatal.

5.4.11 System Call Exception

Cause

A System Call exception occurs during an attempt to execute the SYSCALL instruction. This exception is not maskable.

Processing

The common exception vector is used for this exception, and the Sys code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the SYSCALL instruction unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction.

If the SYSCALL instruction is in a branch delay slot, the BD bit of the Status register is set to 1; otherwise this bit is cleared.

Servicing

When this exception occurs, control is transferred to the applicable system routine.

To resume execution, the EPC register must be altered so that the SYSCALL instruction does not re-execute; this is accomplished by adding a value of 4 to the EPC register before returning.

If a SYSCALL instruction is in a branch delay slot, interpretation of the branch instruction is required to resume execution.

5.4.12 Breakpoint Exception

Cause

A Breakpoint exception occurs when an attempt is made to execute the BREAK instruction. This exception is not maskable.

Processing

The common exception vector is used for this exception, and the BP code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the BREAK instruction unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction.

If the BREAK instruction is in a branch delay slot, the BD bit of the Status register is set to 1.

Servicing

When the Breakpoint exception occurs, control is transferred to the applicable system routine. Additional distinctions can be made by analyzing the unused bits of the BREAK instruction (bits 25 to 6), and loading the contents of the instruction whose address the EPC register contains. A value of 4 must be added to the contents of the EPC register to locate the instruction if it resides in a branch delay slot.

To resume execution, the EPC register must be altered so that the BREAK instruction does not re-execute; this is accomplished by adding a value of 4 to the EPC register before returning.

If a BREAK instruction is in a branch delay slot, interpretation of the branch instruction is required to resume execution.

5.4.13 Coprocessor Unusable Exception

Cause

The Coprocessor Unusable exception occurs when an attempt is made to execute a coprocessor instruction for either:

- ✧ a corresponding coprocessor unit that has not been marked usable (Status register bit, CU0 = 0), or
- ✧ CP0 instructions, when the unit has not been marked usable (Status register bit, CU[0] = 0) and the process executes in User or Supervisor mode.

This exception is not maskable.

Processing

The common exception vector is used for this exception, and the CPU code in the ExcCode field of the Cause register is set. The CE bit of the Cause register indicates which of the four coprocessors was referenced.

The EPC register contains the address of the unusable coprocessor instruction unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

The coprocessor unit to which an attempted reference was made is identified by the CE bit of the Cause register. One of the following processing is performed by the handler:

- ✧ If the process is entitled access to the coprocessor, the coprocessor is marked usable and the corresponding state is restored to the coprocessor.
- ✧ If the process is entitled access to the coprocessor, but the coprocessor does not exist or has failed, interpretation of the coprocessor instruction is possible.
- ✧ If the BD bit is set to 1 in the Cause register, the branch instruction must be interpreted; then the coprocessor instruction can be emulated and execution resumed with the EPC register advanced past the coprocessor instruction.
- ✧ If the process is not entitled access to the coprocessor, the kernel reports UNIX SIGILL/ILL_PRIVIN_FAULT (illegal instruction/privileged instruction fault) signal to the current process, but the exception is fatal.

5.4.14 Reserved Instruction Exception

Cause

The Reserved Instruction exception occurs when an attempt is made to execute one of the following instructions:

- ✧ Instruction with an undefined major opcode (bits 31 to 26)
- ✧ SPECIAL instruction with an undefined minor opcode (bits 5 to 0)
- ✧ REGIMM instruction with an undefined minor opcode (bits 20 to 16)
- ✧ 64-bit instructions in 32-bit User or Supervisor mode

64-bit operations are always valid in Kernel mode regardless of the value of the KX bit in the Status register. This exception is not maskable.

Processing

The common exception vector is used for this exception, and the RI code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the reserved instruction unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

All currently defined MIPS ISA instructions can be executed. The process executing at the time of this exception is handed a UNIX SIGILL/ILL_RESOP_FAULT (illegal instruction/reserved operand fault) signal. This error is usually fatal.

5.4.15 Trap Exception

Cause

The Trap exception occurs when a TGE, TGEU, TLT, TLTU, TEQ, TNE, TGEI, TGEUI, TLTI, TLTUI, TEQI, or TNEI instruction results in a TRUE condition. This exception is not maskable.

Processing

The common exception vector is used for this exception, and the Tr code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the trap instruction causing the exception unless the instruction is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

At the time of a Trap exception, the kernel reports the UNIX SIGFPE/FPE_INTOVF_TRAP (floating-point exception/integer overflow) signal to the current process, but the exception is usually fatal.

5.4.16 Integer Overflow Exception

Cause

An Integer Overflow exception occurs when an ADD, ADDI, SUB, DADD, DADDI or DSUB instruction results in a 2’s complement overflow. This exception is not maskable.

Processing

The common exception vector is used for this exception, and the Ov code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the instruction that caused the exception unless the instruction is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

At the time of the exception, the kernel reports the UNIX SIGFPE/FPE_INTOVF_TRAP (floating-point exception/integer overflow) signal to the current process, but the exception is usually fatal.

5.4.17 Watch Exception

Cause

A Watch exception occurs when a load or store instruction references the physical address specified in the WatchLo/WatchHi register. The WatchLo/WatchHi register specify whether a load or store or both could have initiated this exception.

- When the R bit of the WatchLo register is set to 1: Load instruction
- When the W bit of the WatchLo register is set to 1: Store instruction
- When both the R bit and W bit of the WatchLo register are set to 1: Load instruction or store instruction

The CACHE instruction never causes a Watch exception.

The Watch exception is postponed if the EXL bit is set to 1 in the Status register, and Watch is only maskable by setting the EXL bit to 1 in the Status register.

Processing

The common exception vector is used for this exception, and the WATCH code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the load or store instruction that caused the exception unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

The Watch exception is a debugging aid; typically the exception handler transfers control to a debugger, allowing the user to examine the situation. To continue, the Watch exception must be disabled to execute the faulting instruction. The Watch exception must then be reenabled. The faulting instruction can be executed either by the debugger or by setting breakpoints.

5.4.18 Interrupt Exception

Cause

The Interrupt exception occurs when one of the eight interrupt conditions [Note] is asserted. The significance of these interrupts is dependent upon the specific system implementation.

Each of the eight interrupts can be masked by clearing the corresponding bit in the IM field of the Status register, and all of the eight interrupts can be masked at once by clearing the IE bit of the Status register or setting the EXL/ERL bit.

Note: They are 1 timer interrupt, 5 ordinary interrupts, and 2 software interrupts.

Processing

The common exception vector is used for this exception, and the Int code in the ExcCode field of the Cause register is set.

The IP field of the Cause register indicates current interrupt requests. It is possible that more than one of the bits can be simultaneously set (or cleared) if the interrupt is asserted and then deasserted before this register is read.

The EPC register contains the address of the instruction that caused the exception unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

Servicing

If the interrupt is caused by one of the two software-generated exceptions (SW0 or SW1), the interrupt condition is cleared by setting the corresponding Cause register bit to 0.

If the interrupt is caused by hardware, the interrupt condition is cleared by deactivating the corresponding interrupt request signal.

5.5 EXCEPTION HANDLING AND SERVICING FLOWCHARTS

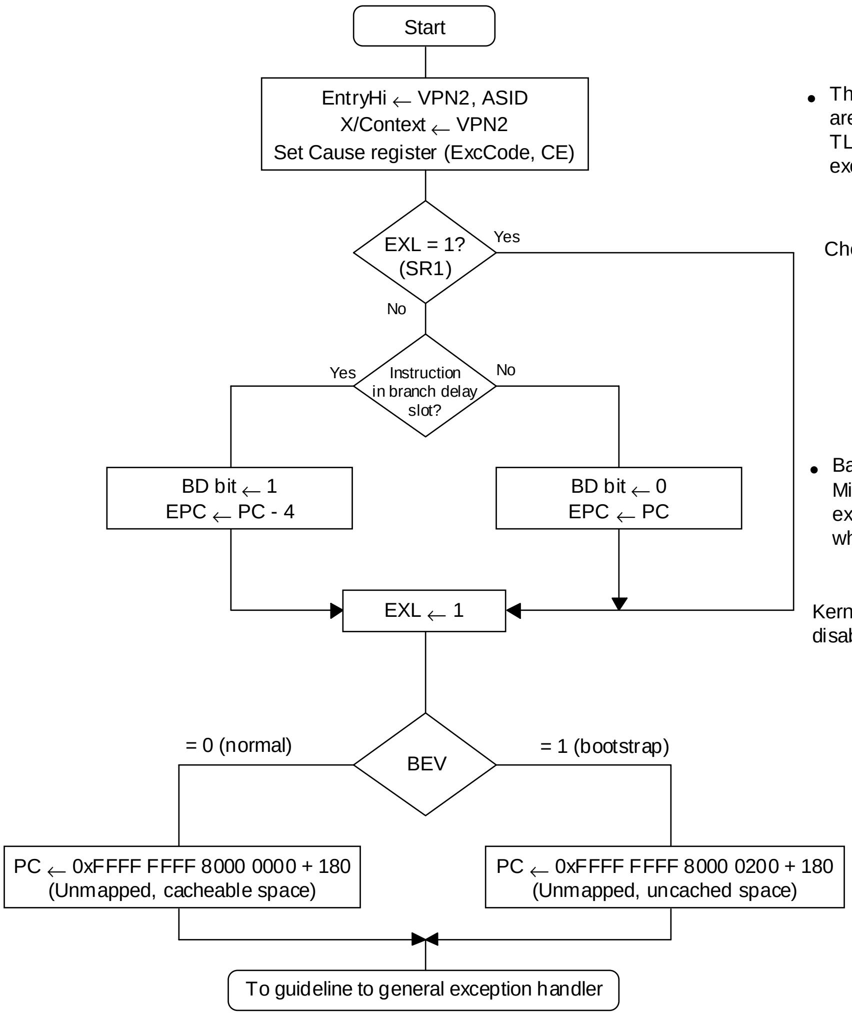
The remainder of this chapter contains flowcharts for the following exceptions and guidelines for their handlers:

- ✧ Common exceptions and a guideline to their exception handler
- ✧ TLB/XTLB Refill exception and a guideline to their exception handler
- ✧ Cache Error exception
- ✧ Cold Reset, Soft Reset and NMI exceptions, and a guideline to their handler.

Generally speaking, the exceptions are “handled” by hardware (HW); the exceptions are then “serviced” by software (SW).

Figure 5-15. Common Exception Handling (1/2)

(a) Handling exceptions other than Cold Reset, Soft Reset, NMI, TLB/XTLB Mismatch, and Cache Error exceptions (hardware)


Start
EntryHi ← VPN2, ASID
X/Context ← VPN2
Set Cause register (ExcCode, CE)
EXL = 1?
(SR1)
Yes
No
Yes
Instruction in branch delay slot?
No
BD bit ← 1
EPC ← PC - 4
BD bit ← 0
EPC ← PC
EXL ← 1
= 0 (normal)
BEV
= 1 (bootstrap)
PC ← 0xFFFF FFFF 8000 0000 + 180
(Unmapped, cacheable space)
PC ← 0xFFFF FFFF 8000 0200 + 180
(Unmapped, uncached space)
To guideline to general exception handler

- The EntryHi and XContext registers are set only when a TLB Mismatch, TLB Invalid, or TLB Modified exception occurs.

Check for multiple exceptions

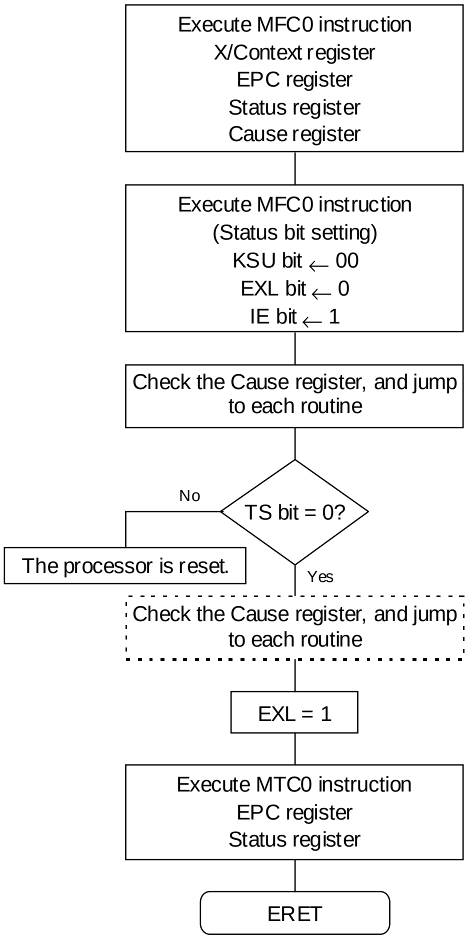
- BadVAddr is set only when a TLB Mismatch, TLB Invalid, or TLB Modified exception occurs. (BadVAddr is not set when a Bus Error exception occurs.)

Kernel mode is set, and interrupts are disabled.

Note The exceptions can be masked by the IE or IM bit. The Watch exception can be set to pending status by setting the EXL bit.

Figure 5-15. Common Exception Handling (2/2)

(b) Servicing common exceptions (software)



- The occurrence of TLB Mismatch, TLB Invalid, and TLB Modified exceptions is disabled by using an unmapped space.
- The occurrence of the Watch and Interrupt exceptions is disabled by setting EXL = 1.
- Other exceptions are avoided in the OS programs.
- However, the Cold Reset, Soft Reset, and NMI exceptions are enabled.

(In Kernel mode, interrupts are enabled.)

- After EXL = 0 is set, all exceptions are enabled (although the Interrupt exception can be masked by the IE and IM bits, and the Cache Error exception can be masked by the DE bit.)

- The register files are saved.

- The execution of the ERET instruction is disabled in the branch delay slots for the other jump instructions.
- The processor does not execute an instruction in the branch delay slot for the ERET instruction.
- PC ← EPC, EXL ← 0

Figure 5-16. TLB/XTLB Refill Exception Handling (1/2)

(a) Handling TLB/XTLB Mismatch exceptions (hardware)

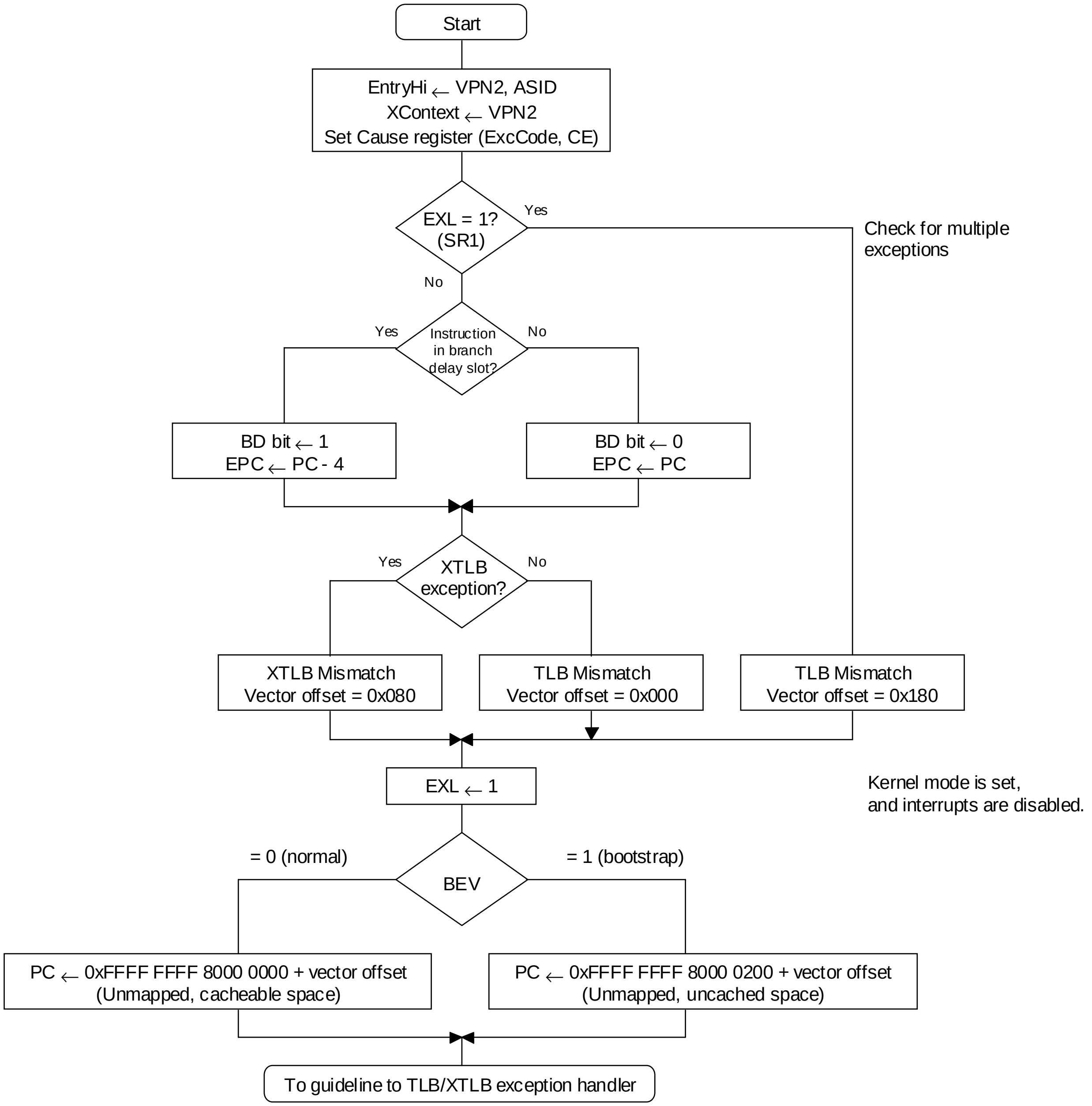
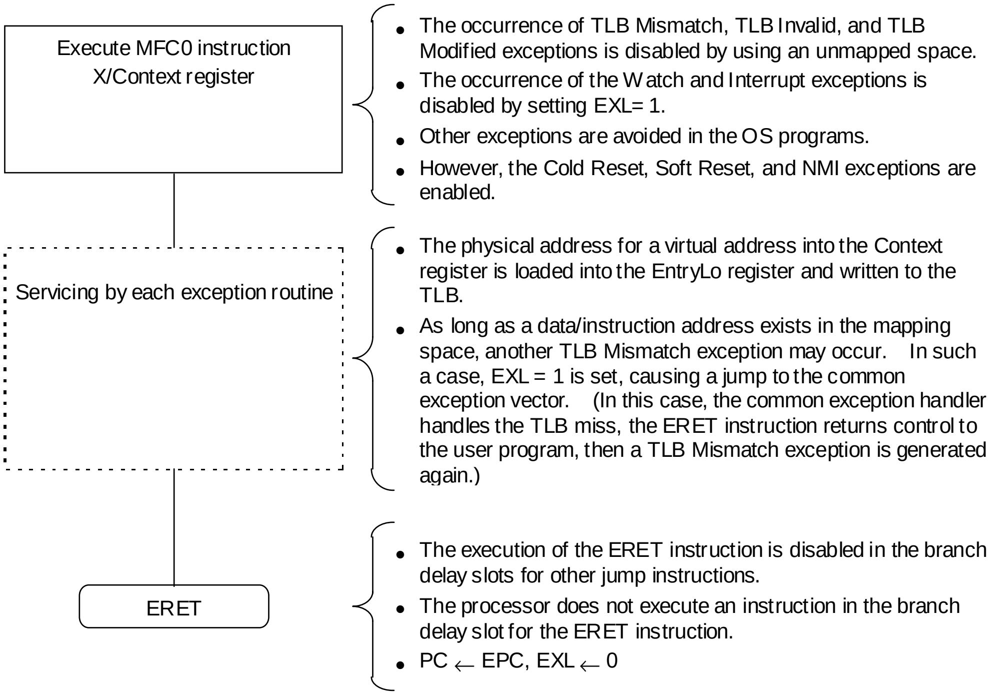


Figure 5-16. TLB/XTLB Refill Exception Handling (2/2)

(b) Servicing TLB/XTLB Mismatch exceptions (software)

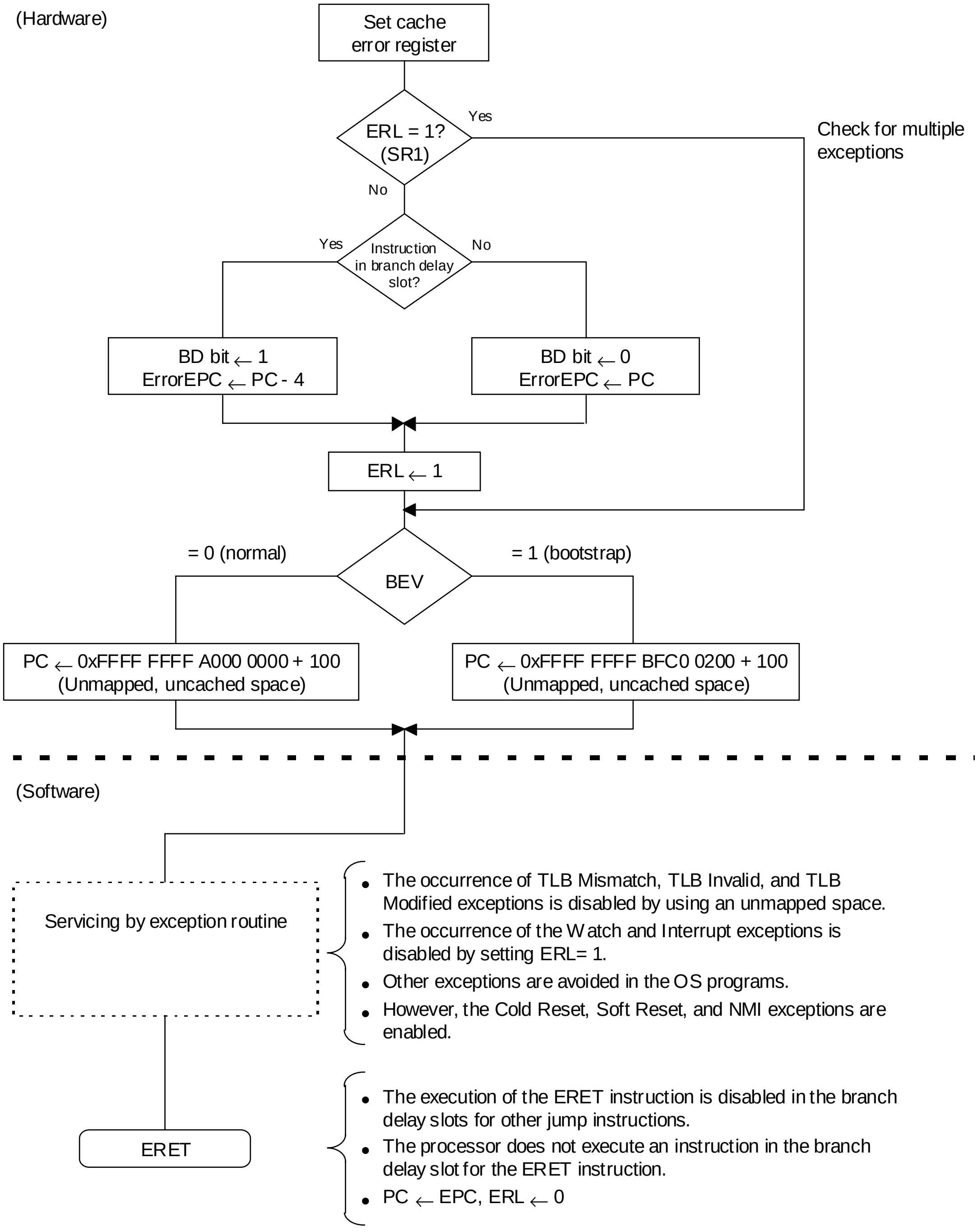


- The occurrence of TLB Mismatch, TLB Invalid, and TLB Modified exceptions is disabled by using an unmapped space.
- The occurrence of the Watch and Interrupt exceptions is disabled by setting EXL= 1.
- Other exceptions are avoided in the OS programs.
- However, the Cold Reset, Soft Reset, and NMI exceptions are enabled.

- The physical address for a virtual address into the Context register is loaded into the EntryLo register and written to the TLB.
- As long as a data/instruction address exists in the mapping space, another TLB Mismatch exception may occur. In such a case, EXL = 1 is set, causing a jump to the common exception vector. (In this case, the common exception handler handles the TLB miss, the ERET instruction returns control to the user program, then a TLB Mismatch exception is generated again.)

- The execution of the ERET instruction is disabled in the branch delay slots for other jump instructions.
- The processor does not execute an instruction in the branch delay slot for the ERET instruction.
- PC ← EPC, EXL ← 0

Figure 5-17. Cache Error Exception Handling

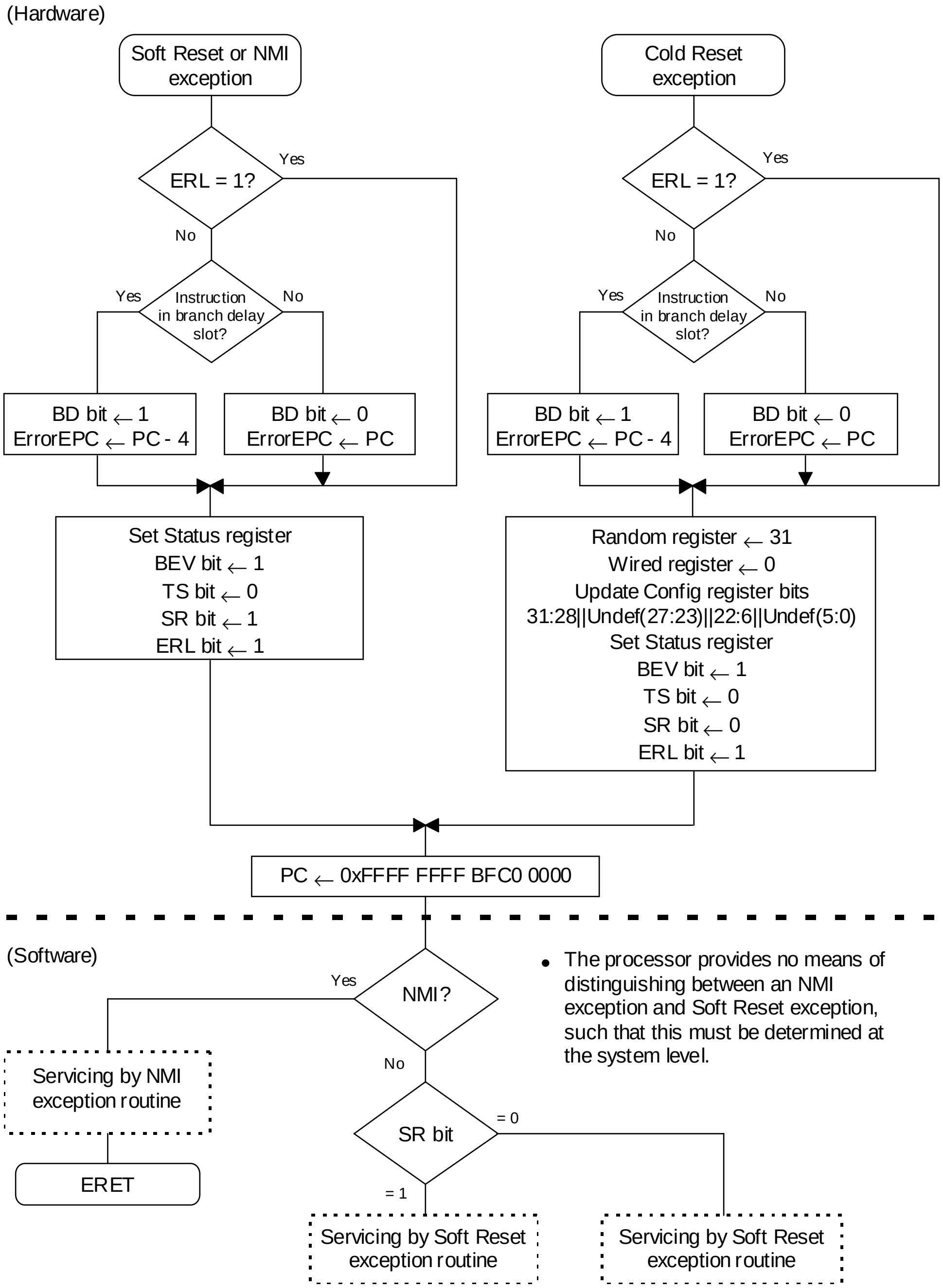

(Hardware)
Set cache error register
ERL = 1? (SR1)
Yes
Check for multiple exceptions
No
Instruction in branch delay slot?
Yes
No
BD bit ← 1
ErrorEPC ← PC - 4
BD bit ← 0
ErrorEPC ← PC
ERL ← 1
= 0 (normal)
BEV
= 1 (bootstrap)
PC ← 0xFFFF FFFF A000 0000 + 100
(Unmapped, uncached space)
PC ← 0xFFFF FFFF BFC0 0200 + 100
(Unmapped, uncached space)
(Software)
Servicing by exception routine
ERET

- The occurrence of TLB Mismatch, TLB Invalid, and TLB Modified exceptions is disabled by using an unmapped space.
- The occurrence of the Watch and Interrupt exceptions is disabled by setting ERL= 1.
- Other exceptions are avoided in the OS programs.
- However, the Cold Reset, Soft Reset, and NMI exceptions are enabled.

- The execution of the ERET instruction is disabled in the branch delay slots for other jump instructions.
- The processor does not execute an instruction in the branch delay slot for the ERET instruction.
- PC ← EPC, ERL ← 0

Note The Cache Error exception can be masked by setting the DE (SR16) bit to 1. When ERL = 1, Cache Error exceptions are masked.

Figure 5-18. Cold Reset, Soft Reset, and NMI Exception Handling



- The processor provides no means of distinguishing between an NMI exception and Soft Reset exception, such that this must be determined at the system level.

[MEMO]

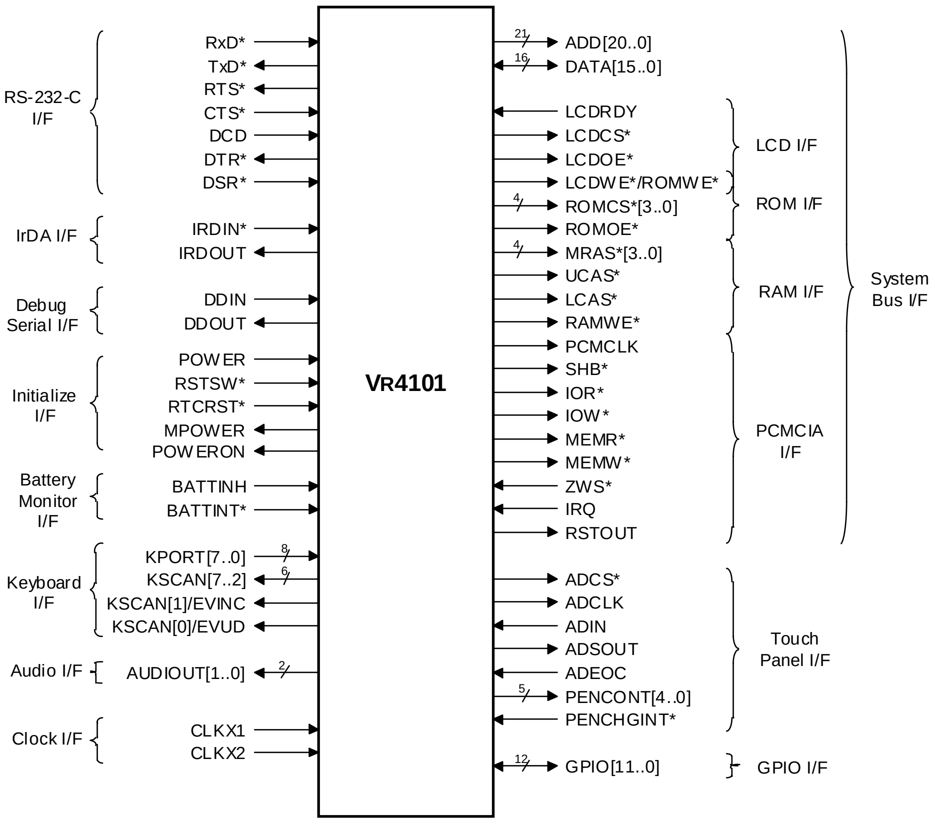
CHAPTER 6 PIN FUNCTIONS

This chapter describes the signals used by and in conjunction with the VR4101 processor.

Low active signals have a trailing asterisk. The signal description also tells if the signal is an input (the processor receives it) or output (the processor sends it out).

Figure 6-1 illustrates the functional groupings of the processor signals.

Figure 6-1. VR4101 Processor Signals



6.1 VR4101 SIGNALS

6.1.1 System Bus Interface Signals

The system bus interface signals are used when the VR4101 processor is connected to the system's DRAM, ROM, LCD, and PCMCIA. Table 6-1 lists the functions of these signals.

Table 6-1. System Bus Interface Signals (1/2)

Signal name	I/O	Definition	Function
ADD[20..0]	O	Address Bus	21-bit address bus. This bus is used to specify DRAM, ROM, LCD, and PCMCIA addresses from the VR4101.
DATA[15..0]	I/O	Data Bus	16-bit data bus. This bus is used to transfer data from the VR4101 to DRAM, ROM, the LCD, and PCMCIA, or vice versa.
LCDCS*	O	LCD Chip Select	LCD chip select signal. This signal becomes active when the VR4101 accesses the LCD by using the ADD bus and DATA bus.
LCDOE*	O	LCD Output Enable	LCD output enable signal. This signal becomes active when the VR4101 accesses the LCD to read data.
LCDWE*/ ROMWE*	O	LCD Write Enable/ ROM Write Enable	Signal generated by multiplexing the LCD write enable signal and FROM write enable signal. This signal functions as the LCD write enable signal when LCDCS is active; the signal becomes active when the VR4101 accesses the LCD to write data. This signal functions as the ROM write enable signal when LCDCS is inactive; the signal becomes active when the VR4101 accesses flash memory to write data.
LCDRDY	I	LCD Ready	LCD ready signal. Activate this signal when the LCD or PCMCIA controller is ready to accept accesses from the VR4101.
ROMCS*[3..0]	O	ROM Chip Select	ROM chip select signal. This signal is used to select a ROM to be accessed from a maximum of four connected ROM chips.
ROMOE*	O	ROM Output Enable	ROM output enable signal. This signal becomes active when the VR4101 accesses ROM to read data.
MRAS*[3..0]	O	DRAM RAS	DRAM RAS signal. This signal becomes active when a valid row address is output on the ADD bus for a RAM chip to be selected for access, from a maximum of four connected RAM chips.
UCAS*	O	Upper CAS	DRAM CAS signal. This signal becomes active when a valid column address is output on the ADD bus when accessing the high-order area in DRAM.
LCAS*	O	Lower CAS	DRAM CAS signal. This signal becomes active when a valid column address is output on the ADD bus when accessing the low-order area in DRAM.
RAMWE*	O	DRAM Write Enable	DRAM write enable signal. This signal becomes active when the VR4101 accesses DRAM to write data.
PCMCLK	O	PCM Clock	PCMCIA card clock. An 8-MHz clock is supplied to the PCMCIA controller.

Table 6-1. System Bus Interface Signals (2/2)

Signal name	I/O	Definition	Function
SHB*	O	System Bus High Byte Enable	PCMCIA bus high-order byte enable signal. This signal becomes active when the high-order byte of the DATA bus is valid for access to PCMCIA.
IOR*	O	I/O Read	PCMCIA card I/O read signal. This signal becomes active when the VR4101 accesses the PCMCIA I/O port to read data.
IOW*	O	I/O Write	PCMCIA card I/O write signal. This signal becomes active when the VR4101 writes data to the PCMCIA I/O port.
MEMR*	O	Memory Read	PCMCIA card memory read signal. This signal becomes active when the VR4101 accesses PCMCIA memory to read data.
MEMW*	O	Memory Write	PCMCIA card memory write signal. This signal becomes active when the VR4101 accesses PCMCIA memory to write data.
ZWS*	I	Zero Wait State	PCMCIA zero wait state signal. Activate this signal when the PCMCIA controller is ready to accept accesses from the VR4101.
IRQ	I	Interrupt Request	PCMCIA card IRQ signal. By asserting this pin, the PCMCIA controller sends an interrupt request to the VR4101.
RSTOUT	O	PCM Reset	PCMCIA card reset signal. This signal becomes active when the VR4101 resets the PCMCIA controller.

6.1.2 Clock Interface Signals

The clock interface signals are used to supply a 32-kHz clock. Table 6-2 lists the functions of these signals.

Table 6-2. Clock Interface Signals

Signal name	I/O	Definition	Function
CLKX1	I	Clock X1	32-kHz clock input pin. This pin is used to connect a 32-kHz crystal.
CLKX2	I	Clock X2	32-kHz clock input pin. This pin is used to connect a second 32-kHz crystal.
V_{DD}P	-	V_{DD} for PLL	Quiet V_{DD} for internal PLL circuit.
GNDP	-	GND for PLL	Quiet GND for internal PLL circuit.

6.1.3 Battery Monitor Interface Signals

The battery monitor interface signals are used by the external circuitry to indicate whether the power required to enable system operation is being supplied. Table 6-3 indicates the functions of the battery monitor interface signals.

Table 6-3. Battery Monitor Interface Signals

Signal name	I/O	Definition	Function
BATTINH	I	Battery Inhibit	Interrupt signal for indicating low battery charge at power-on. The external circuitry checks the battery charge level at power-on, and asserts this pin when it is confirmed that the battery voltage is sufficient to enable operation.
BATTINT*	I	Battery Interrupt	Interrupt signal for indicating an insufficient battery charge level during normal operation. The external circuitry monitors the battery charge level, and asserts this pin when it is confirmed that the battery charge has fallen to a level where the voltage required for operation is longer being supplied.

6.1.4 Initialization Interface Signals

The initialization interface signals are used when an external circuit initializes the processor operating parameters. Table 6-4 lists the functions of these signals.

Table 6-4. Initialization Interface Signals

Signal name	I/O	Definition	Function
MPOWER	O	Main Power On	Signal for turning on the main power. By asserting this pin, the VR4101 turns on the power to the external DC/DC converter.
POWERON	O	Power On State	Signal for indicating that the VR4101 is being activated in Hibernate mode. This signal becomes active upon the detection of an activation factor. It subsequently becomes inactive upon the completion of battery checking.
POWER	I	Power On Switch	Signal for indicating that the power-on switch has been pressed. When the power-on switch is pressed, the external circuitry asserts this pin.
RSTSW*	I	Reset Switch	Signal for indicating that the reset switch has been pressed. When the reset switch is pressed, the external circuitry asserts this pin.
RTCRST*	I	Real-time Clock Reset	Signal for resetting RTC. When the power to the device is turned on for the first time, the external circuitry asserts this pin for about 230 ms.

6.1.5 RS-232-C Interface Signals

The RS-232-C interface signals control data transfer between the VR4101 and the RS-232-C controller. Table 6-5 lists the functions of these signals.

Table 6-5. RS-232-C Interface Signals

Signal name	I/O	Definition	Function
RxD*	I	Receive Data	Send data signal. This signal is used when serial data is transferred from the VR4101 to the RS-232-C driver/receiver.
TxD*	O	Transmit Data	Receive data signal. This signal is used when serial data is transferred from the RS-232-C driver/receiver to the VR4101.
RTS*	O	Request to Send	Send request signal. The VR4101 asserts this signal to send serial data.
CTS*	I	Clear to Send	Send enable signal. The RS-232-C driver/receiver asserts this signal while the controller can accept transferred serial data.
DCD	I	Data Carrier Detection	Carrier detection signal. This signal is asserted while valid serial data is being received.
DTR*	O	Data Terminal Ready	Terminal ready signal. The VR4101 asserts this signal while the VR4101 can both send and receive serial data.
DSR*	I	Data Set Ready	Data set ready signal. Assert this signal when the RS-232-C driver/receiver can transfer serial data to and from the VR4101.

6.1.6 IrDA Interface Signals

The IrDA interface signals control the transfer of data between the VR4101 and IrDA controller. Table 6-6 lists the functions of these signals.

Table 6-6. IrDA Interface Signals

Signal name	I/O	Definition	Function
IRDIN*	I	IrDA Data In	IrDA serial data input signal. This signal is used to transfer data from the VR4101 to the IrDA controller.
IRDOUT	O	IrDA Data Out	IrDA serial data output signal. This signal is used to transfer data from the IrDA controller to the VR4101.

6.1.7 Debug Serial Interface Signals

The debug serial interface signals control data transfer between the VR4101 and debug serial controller. Table 6-7 lists the functions of these signals.

Table 6-7. Debug Serial Interface Signals

Signal name	I/O	Definition	Function
DDIN	I	Debug Serial Data In	Debug serial data input signal. This signal is used to transfer data from the VR4101 to the external debug serial controller.
DDOUT	O	Debug Serial Data Out	Debug serial data output signal. This signal is used to transfer data from the external debug serial controller to the VR4101.

6.1.8 Keyboard Interface Signals

The keyboard interface signals control the keyboard circuit connected to the VR4101. Table 6-8 lists the functions of these signals.

Table 6-8. Keyboard Interface Signals

Signal name	I/O	Definition	Function
KPORT[7..0]	I	Key Scan Data In	Keyboard scan data input signal. This signal is used to scan the pressed/released states of the keyboard.
KSCAN[7..2]	O	Key Scan Data Out	Keyboard scan data output signal. This signal is used to activate the scan lines when the keyboard pressed/released states are being scanned.
KSCAN[1]/ EVINC	O	Key Scan Data Out/ Electric Volume Input Clock	Signal generated by multiplexing the keyboard scan data output signal and electronic volume control clock signal. When EVINC is enabled by the EVINCEN bit in the EVVOLREG, this signal functions as the clock output pin for the electronic volume controller.
KSCAN[0]/ EVUD	O	Key Scan Data Out/ Electric Volume Up/Down	Signal generated by multiplexing the keyboard scan data output signal and electronic volume up/down signal. When EVUD is enabled by the EVUDEN bit in the EVVOLREG, this signal functions as the volume up/down pin for the electronic volume controller.

6.1.9 Audio Interface Signal

The audio interface signal is used to output an audio signal, usually when a WAVE file is being regenerated. Table 6-9 indicates the function of this signal.

Table 6-9. Audio Interface Signal

Signal name	I/O	Definition	Function
AUDIOUT[1..0]	O	Audio Out	Audio output signal. When a WAVE file is regenerated, an audio signal is output.

6.1.10 Touch Panel Interface Signals

The touch panel interface signals control the A/D converter that is connected to the VR4101 and which is used for the touch panel. Table 6-10 lists the functions of these signals.

Table 6-10. Touch Panel Interface Signals

Signal name	I/O	Definition	Function
ADCS*	O	A/D Converter Chip Select	A/D converter chip select signal. This signal is activated when data is transferred to and from the A/D converter.
ADCLK	O	A/D Converter Clock	A/D converter clock output. A clock is supplied to the A/D converter.
ADIN	I	A/D Converter Data In	A/D converter data input signal for receiving A/D converter output data.
ADSOUT	O	A/D Converter Serial Out	A/D converter serial data output signal. This signal is used to output serial data for setting the A/D converter.
ADEOC	I	A/D Converter End Of Change	A/D converter data conversion end signal. Assert this pin to terminate A/D conversion by the A/D converter.
PENCONT [4..0]	O	Touch Panel Control	Touch panel control signal. This signal is output, for example, to control the voltage applied to the touch panel.
PENCHGINT*	I	Pen Change Interrupt	T/P interrupt signal. When a key of the touch panel is pressed, the external circuitry asserts this pin.

6.1.11 General-purpose I/O Signals

These are general-purpose I/O terminals for the VR4101. Among the 12 general-purpose I/O terminals of the VR4101, GPIO[9] is specified for battery cover lock detection interrupt. Table 6-11 shows the functions of these signals.

Table 6-11. General-purpose I/O Signals

Signal name	I/O	Definition	Function
GPIO[11..10]	I/O	GPIO[11..10]	General-purpose I/O terminals.
GPIO[9]	I/O	Battery Lock Detect	Battery cover lock detection signal.
GPIO[8..0]	I/O	GPIO[8..0]	General-purpose I/O terminals.

6.2 STATUS OF PINS UPON A SPECIFIC STATE

Table 6-12 lists the status of the pins upon reset of the VR4101, as well as the status in each power mode.

Table 6-12. Status of Pins upon a Reset (1/2)

Signal name	When reset by RTCRST	When reset by Deadman's SW or RSTSW	In Suspend mode	In Hibernate mode or upon shutdown by HAL timer
ADD[20..0]	0	X	X	0
DATA[15..0]	0	X	X	0
LCDCS*	Hi-Z	1	1	Hi-Z
LCDOE*	Hi-Z	1	1	Hi-Z
LCDWE*/ROMWE*	Hi-Z	1	1	Hi-Z
LCDRDY	Hi-Z	Hi-Z	Hi-Z	Hi-Z
ROMCS*[3..0]	Hi-Z	1	1	Hi-Z
ROMOE*	Hi-Z	1	1	Hi-Z
MRAS*[3..0]	1	1	0	0
UCAS*	1	1	0	0
LCAS*	1	1	0	0
RAMWE*	1	1	1	1
PCMCLK	0	X	X	0
SHB*	0	X	X	0
IOR*	Hi-Z	1	1	Hi-Z
IOW*	Hi-Z	1	1	Hi-Z
MEMR*	Hi-Z	1	1	Hi-Z
MEMW*	Hi-Z	1	1	Hi-Z
ZWS*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
IRQ	Hi-Z	Hi-Z	Hi-Z	Hi-Z
RSTOUT	Hi-Z	0	**Note**	Hi-Z
CLKX1	Hi-Z	Hi-Z	Hi-Z	Hi-Z
CLKX2	Hi-Z	Hi-Z	Hi-Z	Hi-Z
BATTINH	Hi-Z	Hi-Z	Hi-Z	Hi-Z
BATTINT*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
MPOWER	0	1	1	0
POWERON	0	0	0	0
POWER	Hi-Z	Hi-Z	Hi-Z	Hi-Z
RSTSW*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
RTCRST*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
RxD*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
TxD*	1	1	**Note**	1
RTS*	1	1	**Note**	1
CTS*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
DCD	Hi-Z	Hi-Z	Hi-Z	Hi-Z
DTR*	1	1	**Note**	1
DSR*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
IRDIN*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
IRDOUT	Hi-Z	Hi-Z	**Note**	Hi-Z
DDIN	Hi-Z	Hi-Z	Hi-Z	Hi-Z
DDOUT	1	1	**Note**	1

Note Retains the status existing in the previous Fullspeed mode.

Remark 0: outputs low level, 1: outputs high level, Hi-Z: high-impedance, X: undefined.

Table 6-12. Status of Pins upon a Reset (2/2)

Signal name	When reset by RTCRST	When reset by Deadman's SW or RSTSW	In Suspend mode	In Hibernate mode or upon shutdown by HAL timer
KPORT[7..0]	Hi-Z	Hi-Z	Hi-Z	Hi-Z
KSCAN[7..2]	Hi-Z	Hi-Z	**Note**	Hi-Z
KSCAN[1]/EVINC	Hi-Z	Hi-Z	**Note**	Hi-Z
KSCAN[0]/EVUD	Hi-Z	Hi-Z	**Note**	Hi-Z
AUDIOUT[1..0]	0	0	0	0
ADCS*	Hi-Z	Hi-Z	**Note**	Hi-Z
ADCLK	0	0	**Note**	0
ADIN	Hi-Z	Hi-Z	Hi-Z	Hi-Z
ADSOUT	0	0	**Note**	0
ADEOC	Hi-Z	Hi-Z	Hi-Z	Hi-Z
PENCONT[4..0]	Hi-Z	Hi-Z	**Note**	Hi-Z
PENCHGINT*	Hi-Z	Hi-Z	Hi-Z	Hi-Z
GPIO[11..0]	Hi-Z	Hi-Z	**Note**	Hi-Z

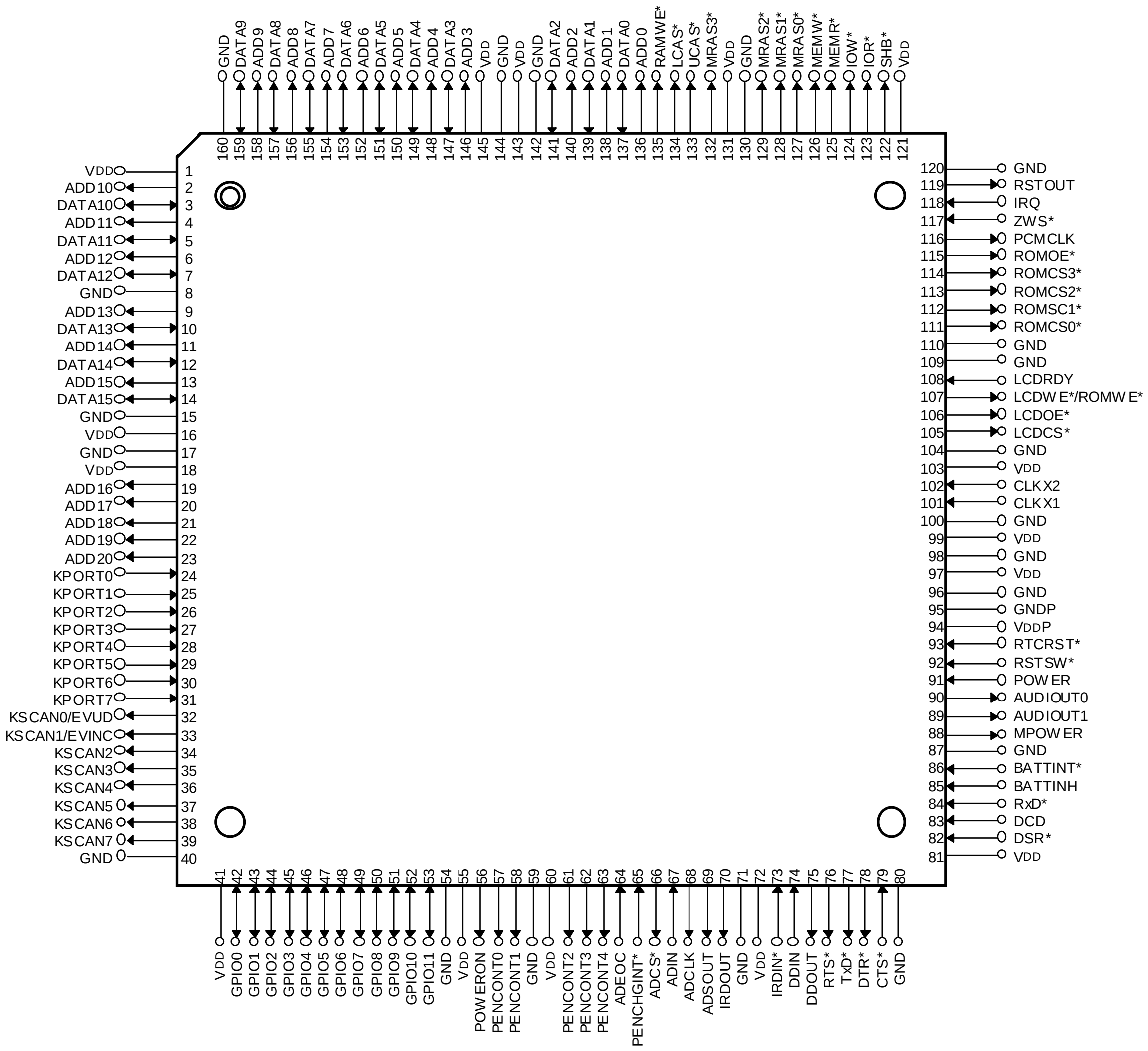
Note Retains the status existing in the previous Fullspeed mode.

Remark 0: outputs low level, Hi-Z: high-impedance.

6.3 PIN CONFIGURATION

This section shows the pinouts and pin assignment for the VR4101.

Figure 6-2. Pinout of the 160-pin LQFP



[MEMO]

CHAPTER 7 INITIALIZATION INTERFACE

This chapter describes the Initialization interface and the processor modes. This includes the reset signal description and types, and initialization sequence, with signals and timing dependencies, and the user-selectable processor modes.

7.1 RESET FUNCTION

Five methods of resetting the VR4101 processor are supported. Each of these methods is outlined below.

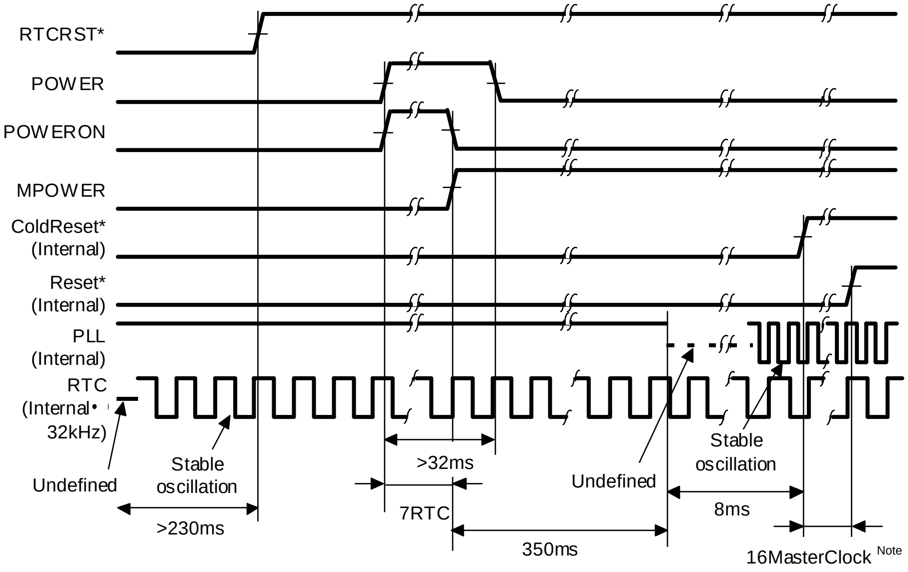
7.1.1 RTC Reset

When turning on the power, activate the RTCRST* pin. After the power supply voltage has stabilized at 2.5 V or more, wait about 230 ms for the 32-kHz oscillator circuit to start, then deactivate the RTCRST* pin. The RTC unit will start counting. Next, after the power supply voltage has stabilized to between 3.0 V and 3.6 V, activate the POWER pin. The VR4101 will assert the POWERON pin then check the battery charge level by means of the BATTINH signal. If the battery charge level is sufficient and the GPIO[9] (BATTLOCK) pin has been asserted, the VR4101 will assert the MPOWER pin to start the external DC/DC converter. After the DC/DC converter has stabilized (after about 350 ms), the VR4101 starts PLL oscillation, and all clocks are started (Note, however, that after the start of PLL oscillation, about 8 ms is required for PLL oscillation to stabilize).

The RTC reset does not save any state information, instead completely initializing the internal states of the processor. Moreover, the processor does not instigate a DRAM transition to self-refresh mode, so that the contents of DRAM after an RTC reset will be unpredictable.

After a reset, the processor assumes system bus mastership, then begins access to a reset vector in the ROM space. Upon a reset, the VR4101 initializes only some of the internal states. So, initialize the processor completely by software.

Figure 7-1. RTC Reset



Note MasterClock is the basic clock used in the CPU core.

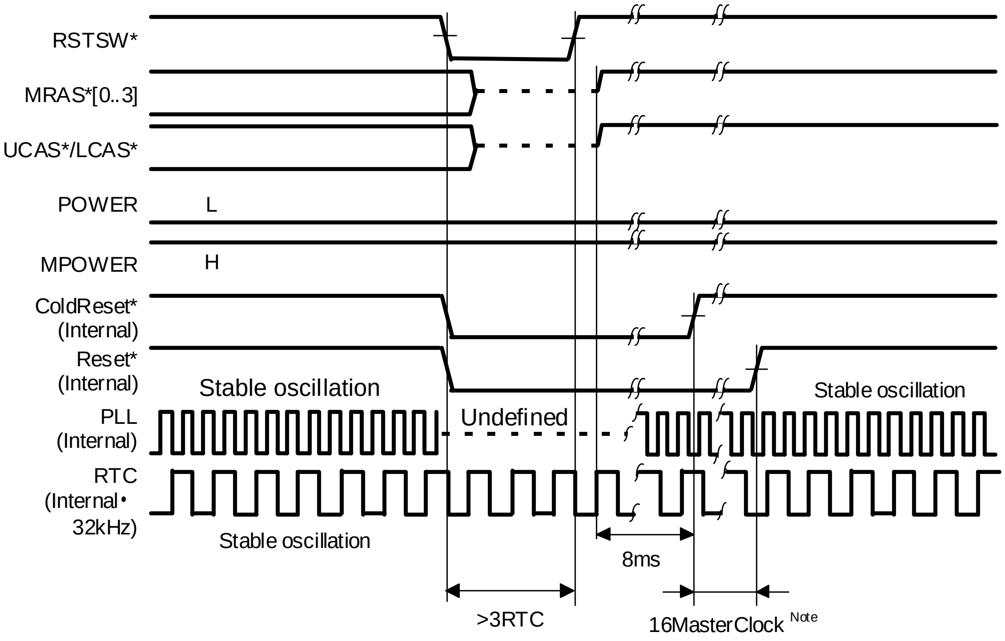
7.1.2 RSTSW

Activate the RSTSW* pin then, after 100 µs, deactivate the RSTSW* pin. The VR4101 immediately starts PLL oscillation, and all clocks are started (Note, however, that after the start of PLL oscillation, about 8 ms is required for PLL oscillation to stabilize).

The RSTSW reset initializes all internal states, except for the RTC timer and PMU. Moreover, the processor does not instigate a DRAM transition to self-refresh mode, so that the contents of DRAM after an RTC reset will be unpredictable.

After a reset, the processor assumes system bus mastership, then begins access to a reset vector in the ROM space. Upon a reset, the VR4101 initializes only some of the internal states. So, initialize the processor completely by software.

Figure 7-2. RSTSW



Note MasterClock is the basic clock used in the CPU core.

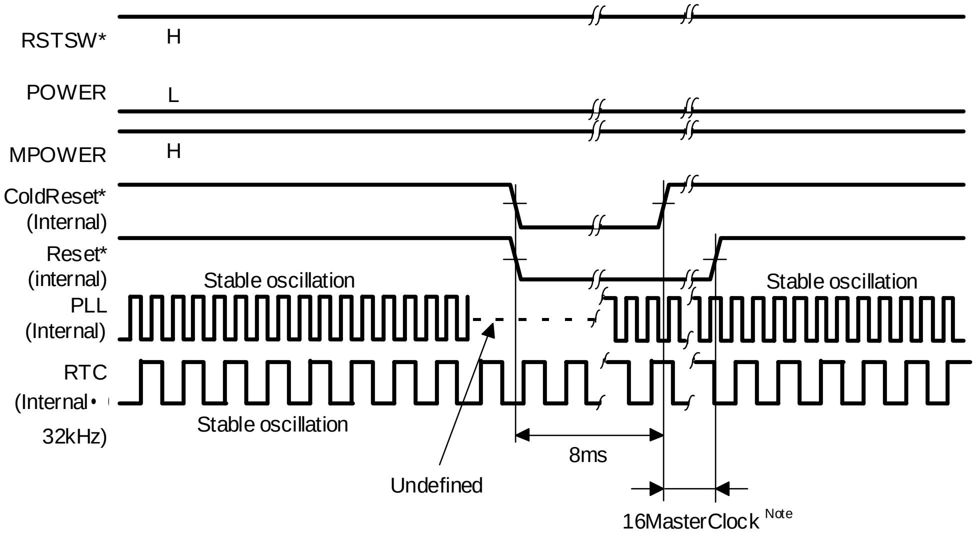
7.1.3 Deadman's SW

If Deadman's SW is not cleared within a specified time after being enables, the VR4101 shifts to the reset state immediately. Deadman's SW is set and cleared by software.

The reset by Deadman's SW initializes all internal states, except for the RTC timer and PMU. Moreover, the processor does not instigate a DRAM transition to self-refresh mode, so that the contents of DRAM after a reset by Deadman's SW will be unpredictable.

After a reset, the processor assumes system bus mastership, then begins access to a reset vector in the ROM space. Upon a reset, the VR4101 initializes only some of the internal states. So, initialize the processor completely by software.

Figure 7-3. Deadman's SW



Note MasterClock is the basic clock used in the CPU core.

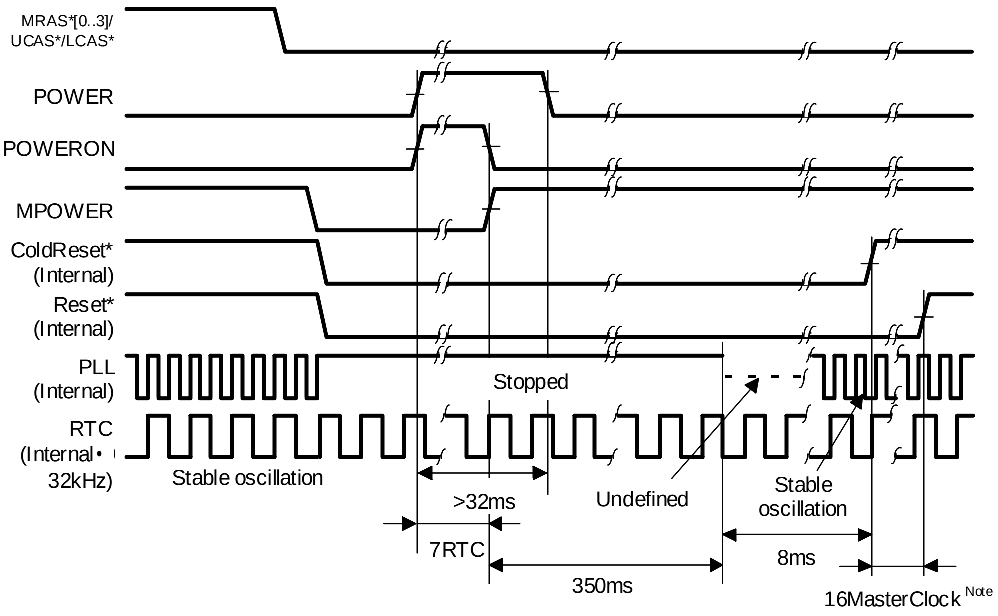
7.1.4 Software Shutdown

When software executes the HIBERNATE instruction, the VR4101 instigates a DRAM transition to self-refresh mode, deactivates the MPOWER pin, then is reset. The VR4101 returns from the reset state when the POWER pin is asserted or the WakeUpTimer interrupt is generated.

The SW Shutdown reset initializes all internal states except for the RTC timer and PMU.

After a reset, the processor assumes system bus mastership, then begins access to a reset vector in the ROM space. Upon a reset, the VR4101 initializes only some of the internal states. So, initialize the processor completely by software.

Figure 7-4. Software Shutdown



Note MasterClock is the basic clock used in the CPU core.

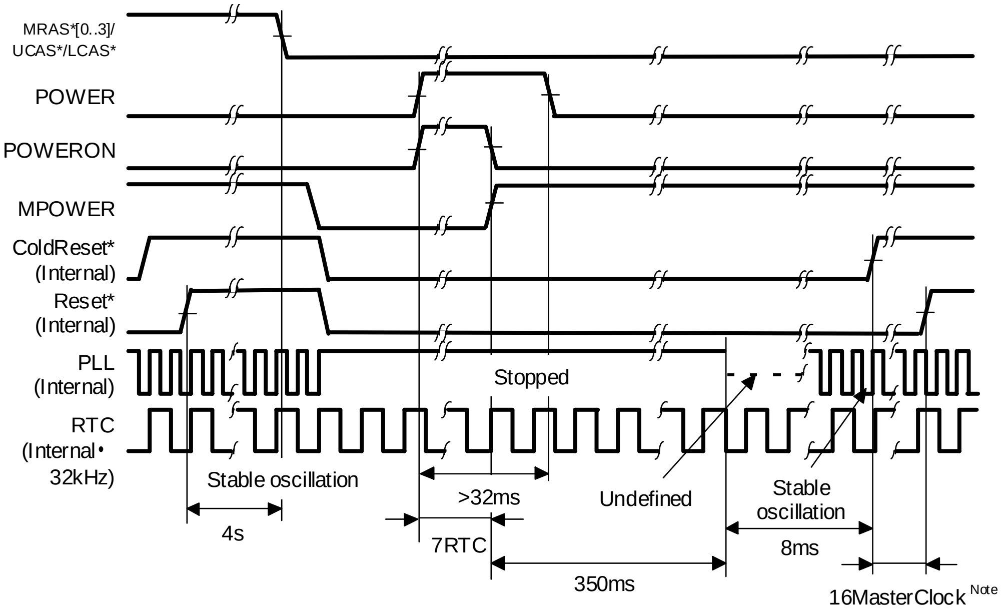
7.1.5 HALTimer Shutdown

If HALTimer is not cleared by software within about 4 seconds of the RTC reset being cleared (if the HALTIMERRST bit of the PMUCNTREG is not set to 1), the VR4101 is reset. The VR4101 returns from the reset state when the POWER pin is asserted or the WakeUpTimer interrupt is generated.

The HALTimer reset initializes all internal states except for the RTC timer and PMU.

After a reset, the processor assumes system bus mastership, then begins access to a reset vector in the ROM space. Upon a reset, the VR4101 initializes only some of the internal states. So, initialize the processor completely by software.

Figure 7-5. HALTimer Shutdown



Note MasterClock is the basic clock used in the CPU core.

7.2 POWER-ON SEQUENCE

An activation factor causes the VR4101 to transit from Hibernate or Shutdown mode to Fullspeed mode. As activation factors, the following are supported: asserting the POWERON pin, asserting the DCD pin, and using the WakeUp timer. Once an activation factor has been detected, the VR4101 asserts the POWERON pin to notify the external circuit that power is being supplied to the VR4101. Three RTC clocks after the POWERON pin has been asserted, the VR4101 checks the states of the BATTINH and GPIO[9] (BATTLOCK) pins. If the BATTINH or GPIO[9] (BATTLOCK) pin is low, the POWERON pin is negated one RTC clock after the check, such that the VR4101 is not activated. If both the BATTINH and GPIO[9] (BATTLOCK) pins are high, the POWERON pin is negated four RTC clocks after the check. Then, the MPOWER pin is asserted, after which the VR4101 is activated.

Figure 7-6 shows an example timing chart where the VR4101 is activated normally. Figure 7-7 shows an example timing chart where the VR4101 is not activated because of the BATTINH pin being low.

See Chapter 15 for more details about power-on sequence for each activation factor.

Figure 7-6. VR4101 Activation Sequence (When Activated Normally)

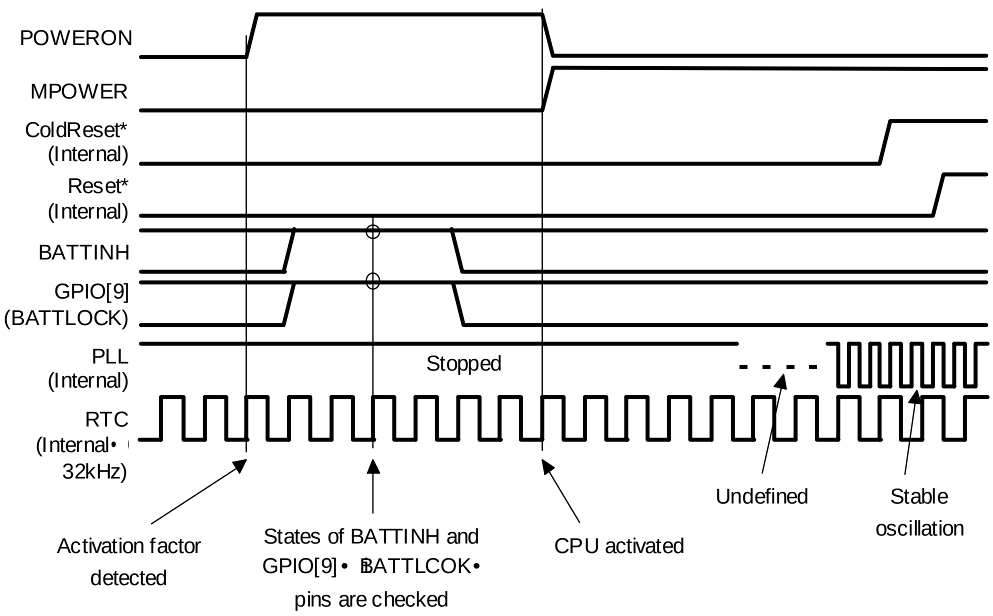
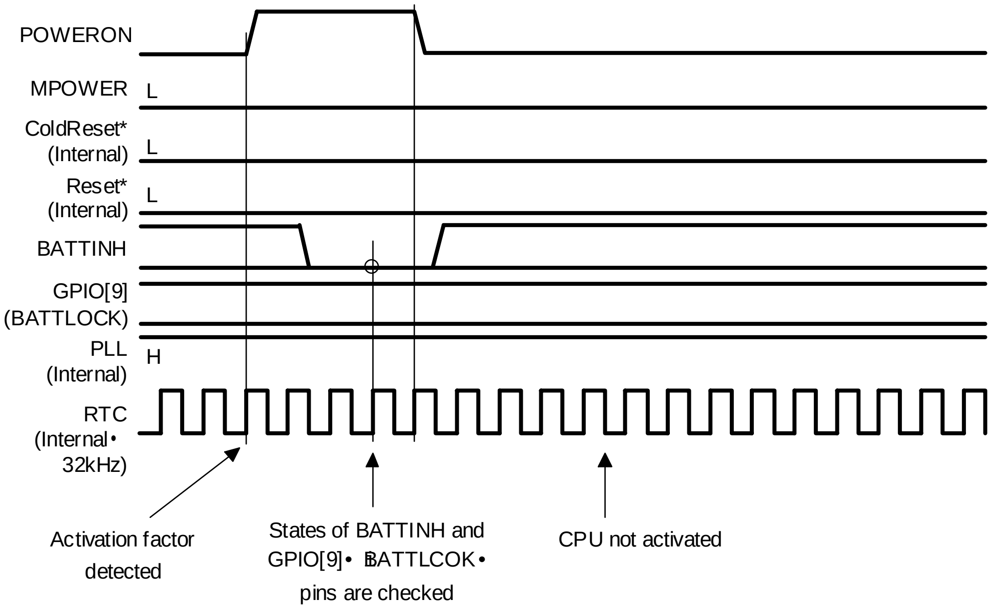


Figure 7-7. VR4101 Activation Sequence (When Activation Fails)



7.3 RESET OF THE CPU CORE

This section describes the reset sequence of the VR4100 CPU core. For details about factors of reset or reset of the whole VR4101, refer to 7.1 and Chapter 15.

7.3.1 Cold Reset

A Cold Reset completely initializes the CPU core, except for the following register bits.

- ✧ The TS and SR bits of the Status register are cleared to 0.
- ✧ The ERL and BEV bits of the Status register are set to 1.
- ✧ The upper limit value (31) is set in the Random register.
- ✧ The Wired register is initialized to 0.
- ✧ Bits 31 to 28 of the Config register are set to 0 and bits 22 to 3 to 0x04800; the other bits are undefined.
- ✧ The values of the other registers are undefined.

Once power to the processor is established, the ColdReset* (internal) and the Reset* (internal) signals are asserted and a Cold Reset is started. After approximately 2 ms assertion, the ColdReset* signal is deasserted synchronously with MasterOut. Then the Reset* signal is deasserted synchronously with MasterOut, and the Cold Reset is completed.

Upon reset, the CPU core becomes bus master and drives the SysAD bus (internal). After Reset* is deasserted, the CPU core branches to the Reset exception vector and begins executing the reset exception code.

7.3.2 Soft Reset

Caution Soft Reset is not supported in the present VR4101.

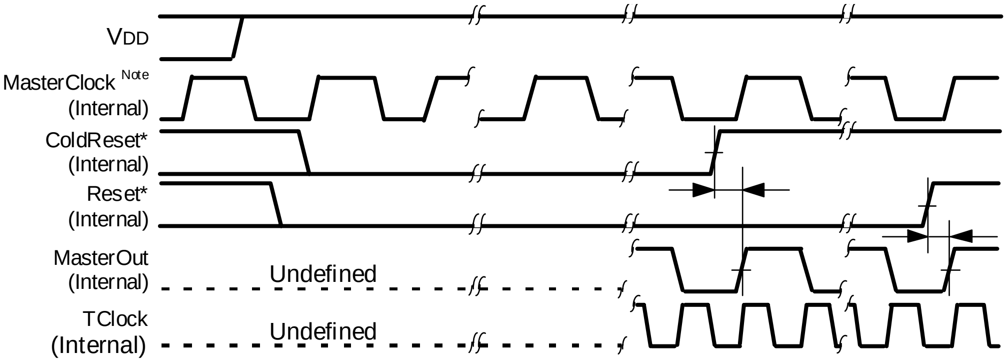
A Soft Reset initializes the CPU core without affecting the clocks; in other words, a Soft Reset is a logic reset. In a Soft Reset, the CPU core retains as much state information as possible; all state information except for the following is retained:

- The TS bit of the Status register is cleared to 0.
- The SR, ERL and BEV bits of the Status register are set to 1.
- The Count register is initialized to 0.
- The IP7 bit of the Cause register is cleared to 0.
- Any Interrupts generated on the SysAD bus are cleared.
- NMI is cleared.
- The Config register is initialized.

A Soft Reset is started by assertion of the Reset* signal, and is completed at the deassertion of the Reset* signal synchronized with MasterOut. In general, data in the CPU core is preserved for debugging purpose.

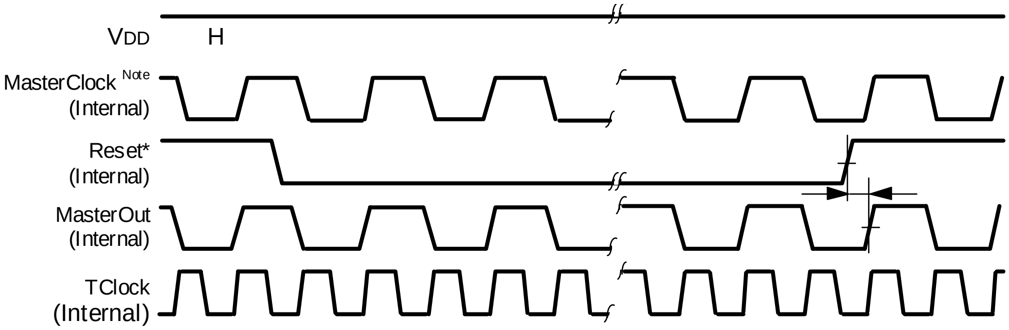
Upon reset, the CPU core becomes bus master and drives the SysAD bus (internal). After Reset* is deasserted, the CPU core branches to the Reset exception vector and begins executing the reset exception code.

Figure 7-8. Cold Reset



Note MasterClock is the basic clock used in the CPU core.

Figure 7-9. Soft Reset



Note MasterClock is the basic clock used in the CPU core.

7.4 VR4101 PROCESSOR MODES

The VR4101 processor supports several user-selectable modes. The CPU core mode is set by writing to the Status register and Config register. The built-in peripheral circuit mode is set by writing to the I/O register.

This section describes the operating modes of the CPU core. For a description of the operating modes of the built-in peripheral circuitry, see the relevant chapters.

7.4.1 Power Modes

The VR4101 supports four power modes: Fullspeed, Standby, Suspend, and Hibernate modes.

(1) Fullspeed mode

Normally the processor clock (PClock) operates at 33 MHz. The system bus clock operates at the same speed as the PClock.

By default, the Fullspeed mode is used. The processor returns to the Fullspeed mode after any reset.

(2) Standby mode

When a STANDBY instruction is executed, the processor is placed in Standby mode. In Standby mode, all internal clocks in the CPU core, excluding the timer and interrupt clocks, are held high. All peripheral units operate in the same way as in Fullspeed mode. This means that DMA operation is also enabled in Standby mode.

When the STANDBY instruction terminates the WB stage, the VR4101 waits for the internal SysAD bus (internal) to become idle. Then, those clocks internal to the CPU core are shut down, causing pipeline operation to terminate. The PLL, timer, interrupt clocks, and internal bus clocks (TClock and MasterOut) continue operation.

Any interrupt, including an internally generated timer interrupt, return the processor placed in Standby mode to Fullspeed mode.

(3) Suspend mode

When a SUSPEND instruction is executed, the processor is placed in Suspend mode. In Suspend mode, the processor stalls the pipeline, and causes all internal clocks in the CPU core, excluding the PLL and interrupt clocks, to be held high. Moreover, the processor stops the supply of TClock to the peripheral units. So, only some peripheral units, such as an interrupt unit (DCD control, etc.), can operate in Suspend mode. In this state, the register data and cache data are preserved.

When the SUSPEND instruction terminates the WB stage, the VR4101 instigates a DRAM transition to self-refresh mode, then waits for the internal SysAD bus (internal) to become idle. Then, those clocks internal to the CPU core are shut down, causing pipeline operation to terminate. Moreover, the supply of TClock to the peripheral units is stopped. However, the PLL, timer, interrupt clocks, and MasterOut continue operation.

The processor remains in Suspend mode until an interrupt is accepted. As soon as an interrupt is accepted, the processor returns to the Fullspeed mode.

(4) Hibernate mode

The users may set the processor to Hibernate mode with HIBERNATE instruction. In the Hibernate mode, the processor quits supplying clocks to all of the units. At the time, the contents of the registers and caches are kept, and TClock and MasterOut output is stopped. The processor remains in Hibernate mode until the POWER pin is asserted, a WakeUpTimer interrupt is generated, or the DCD pin is asserted. When the POWER pin is asserted, a WakeUpTimer interrupt is generated, or the DCD pin is asserted, the processor returns to Fullspeed mode. In Hibernate mode, the power consumption is slightly more than 0 W (Because of the 32-kHz oscillator and built-in peripheral circuits that operate at 32 kHz, the power consumption can never fall completely to 0 W).

7.4.2 Privilege Modes

The VR4101 supports three modes of system privilege: kernel, supervisor, and user extended addressing. This section describes these three modes.

(1) Kernel extended addressing mode

If the KX bit in the Status register is set, it enables MIPS III opcodes in Kernel mode and causes the TLB mismatch on kernel addresses to use the Extended TLB Mismatch exception vector.

(2) Supervisor extended addressing mode

If the SX bit in the Status register is set, it enables MIPS III opcodes in Supervisor mode and causes the TLB mismatch on supervisor addresses to use the Extended TLB Mismatch exception vector.

(3) User extended addressing mode

If the UX bit in the Status register is set, it enables MIPS III opcodes in User mode and causes the TLB mismatch on user addresses to use the Extended TLB Mismatch exception vector. If the bit is clear, it enables MIPS I and II opcodes and 32-bit virtual address.

7.4.3 Reverse Endianess

When the RE bit in the Status register is set, endianess as seen by user software is reversed. However, the RE bit in the Status register must be set to 0 since the VR4101 supports the little-endian order only.

7.4.4 Bootstrap Exception Vector (BEV)

This bit is used when diagnostic tests cause exceptions to occur prior to verifying proper operation of the cache and main memory system.

This bit is automatically set to 1 at reset and NMI exception.

When set, the BEV bit in the Status register causes the TLB Mismatch exception vector to be relocated to a virtual address of 0xFFFF FFFF BFC0 0200 and the common exception vector relocated to address 0xFFFF FFFF BFC0 0380.

When BEV is cleared, these vectors are located at 0xFFFF FFFF 8000 0000 (TLB Mismatch) and 0xFFFF FFFF 8000 0180 (common).

7.4.5 Cache Error Check

When a store instruction is executed with the CE bit of the Status register set, the contents of the Parity Error register can be written to the parity bit positions of the data cache, instead of the parity generated by the store instruction. When a CACHE instruction with Fill specified is executed, the contents of the Parity Error register can be written to the parity bit positions of the instruction cache, instead of the instruction parity bits.

7.4.6 Disable Parity Errors

When the DE bit in the Status register is set, the processor does not take an exception on a cache parity error.

7.4.7 Interrupt Enable (IE)

When this bit in the Status register is clear, all interrupts other than the reset and the non-maskable interrupt are not allowed.

CHAPTER 8 CACHE ORGANIZATION AND OPERATION

This chapter describes in detail the cache memory: its place in the VR4100 CPU core memory organization, and individual organization of the caches.

This chapter uses the following terminology:

″ The data cache may also be referred to as the D-cache.

″ The instruction cache may also be referred to as the I-cache.

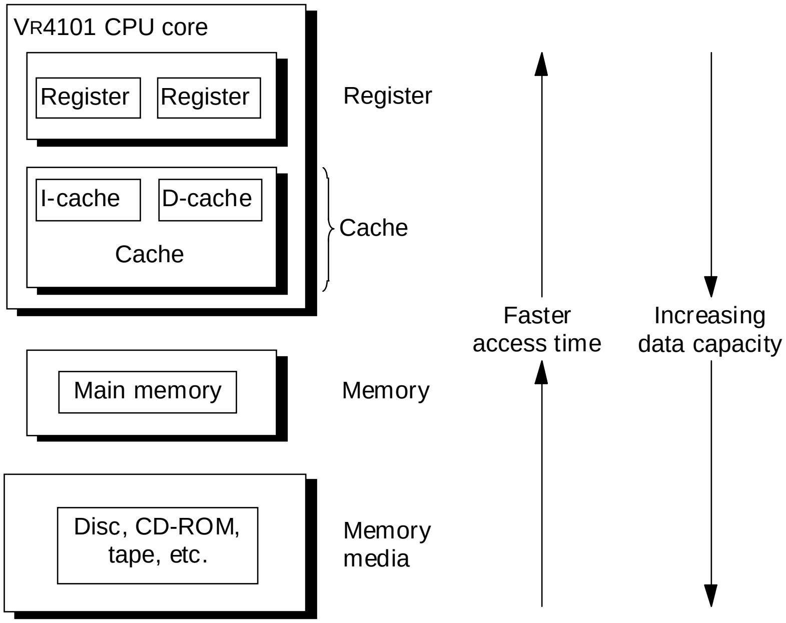
These terms are used interchangeably throughout this book.

8.1 MEMORY ORGANIZATION

Figure 8-1 shows the VR4100 CPU core system memory hierarchy. In the logical memory hierarchy, the caches lie between the CPU and main memory. They are designed to make the speedup of memory accesses transparent to the user.

Each functional block in Figure 8-1 has the capacity to hold more data than the block above it. For instance, physical main memory has a larger capacity than the caches. At the same time, each functional block takes longer to access than any block above it. For instance, it takes longer to access data in main memory than in the CPU on-chip registers.

Figure 8-1. Logical Hierarchy of Memory



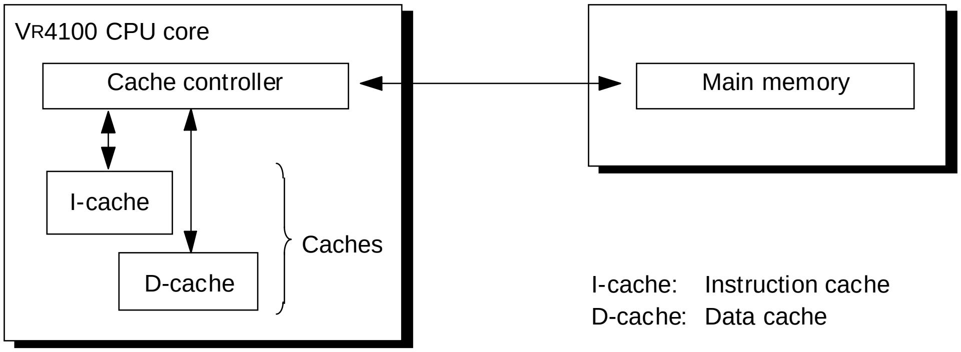
The VR4100 CPU core has two on-chip caches: one holds instructions (the instruction cache), the other holds data (the data cache). The instruction and data caches can be read in one PClock cycle.

Data writes are pipelined and can complete at a rate of one per PClock cycle. In the first stage of the cycle, the store address is translated and the tag is checked; in the second stage, the data is written into the data RAM.

8.2 CACHE ORGANIZATION

This section describes the organization of the on-chip data and instruction caches. Figure 8-2 provides a block diagram of the VR4100 CPU core cache and memory model.

Figure 8-2. Cache Support



(1) Cache Line Lengths

A cache line is the smallest unit of information that can be fetched from main memory for the cache, and that is represented by a single tag.

The line size for the instruction/data cache is 4 words (16 bytes).

(2) Cache Sizes

The instruction cache in the VR4100 CPU core is 2 Kbytes; the data cache is 1 Kbytes.

8.2.1 Organization of the Instruction Cache (I-Cache)

Each line of I-cache data (although it is actually an instruction, it is referred to as data to distinguish it from its tag) has an associated 24-bit tag that contains a 22-bit physical address, a single Valid bit, and a single Parity bit. Word parity is used on I-cache data (1 bit of parity per word).

The VR4100 CPU core I-cache has the following characteristics:

″ direct-mapped

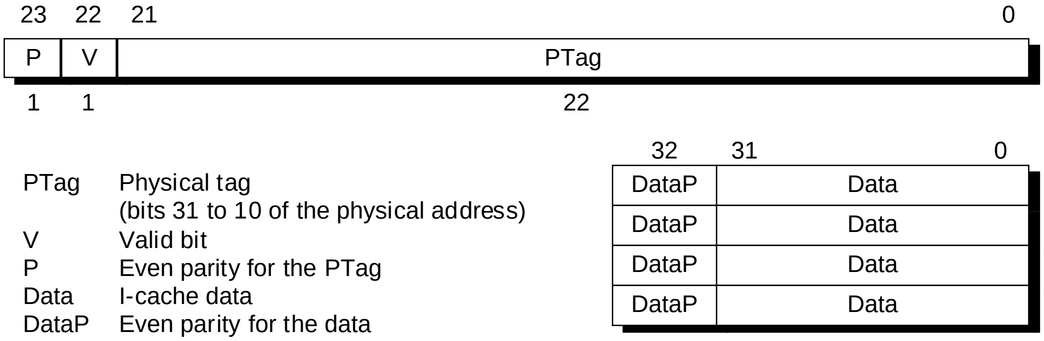
″ indexed with a virtual address

″ checked with a physical tag

″ organized with a 4-word (16-byte) cache line.

Figure 8-3 shows the format of a 4-word (16-byte) I-cache line.

Figure 8-3. Instruction Cache Line Format



PTag	Physical tag (bits 31 to 10 of the physical address)
V	Valid bit
P	Even parity for the PTag
Data	I-cache data
DataP	Even parity for the data

8.2.2 Organization of the Data Cache (D-Cache)

Each line of D-cache data has an associated 26-bit tag that contains a 22-bit physical address, a Valid bit, a Parity bit, a Write-back bit, and a parity bit for Write-back.

The VR4100 CPU core D-cache has the following characteristics :

″ write-back

″ direct-mapped

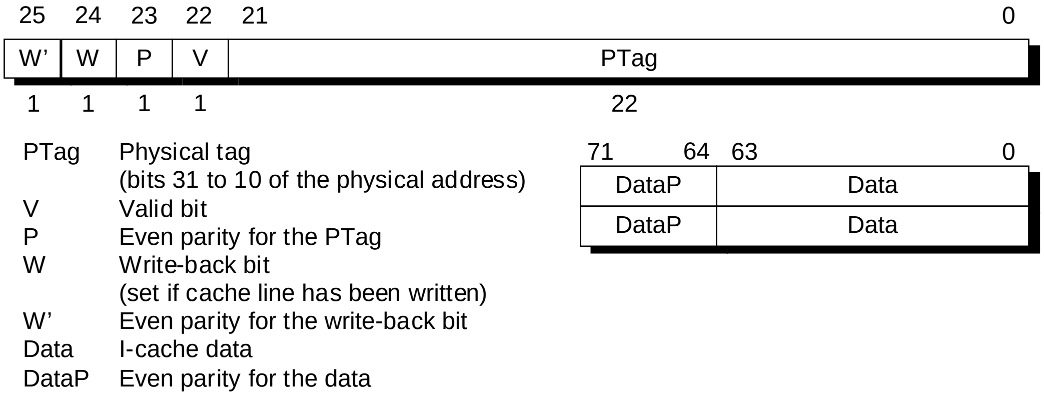
″ indexed with a virtual address

″ checked with a physical tag

″ organized with a 4-word (16-byte) cache line.

Figure 8-4 shows the format of a 4-word (16-byte) D-cache line.

Figure 8-4. Data Cache Line Format



PTag	Physical tag (bits 31 to 10 of the physical address)
V	Valid bit
P	Even parity for the PTag
W	Write-back bit (set if cache line has been written)
W'	Even parity for the write-back bit
Data	I-cache data
DataP	Even parity for the data

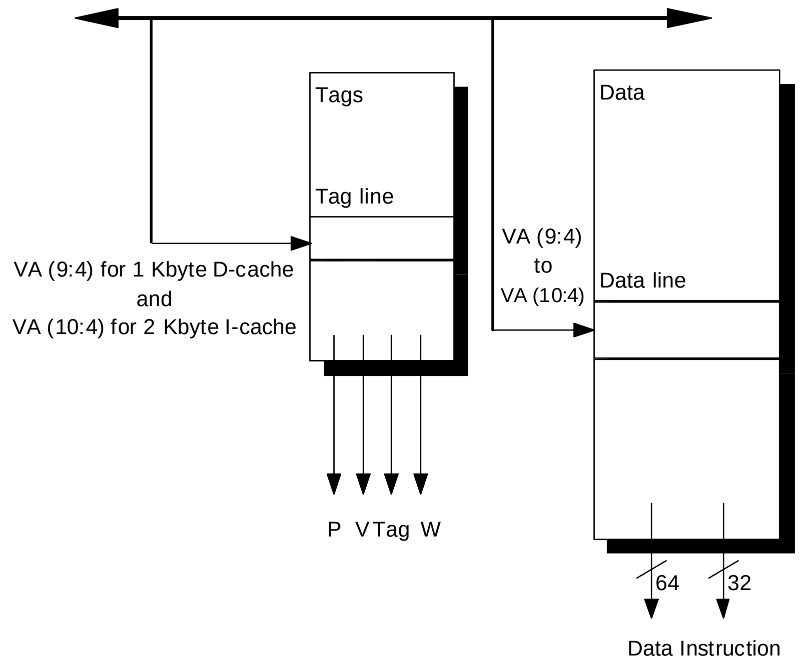
8.2.3 Accessing the Caches

Figure 8-5 shows the virtual address (VA) index into the caches. The number of virtual address bits used to index the instruction and data caches depends on the cache size.

For example, VA (9:4) accesses the 1-Kbyte page tag in the data cache with its 4-word line: VA (9) addresses 1 Kbytes and VA (4) provides quadword resolution.

Similarly, VA (10:4) accesses an 4-word tag in a 2 Kbyte I-cache: VA (4) provides quadword resolution and VA (10) addresses 2 Kbytes.

Figure 8-5. Cache Data and Tag Organization



8.3 CACHE OPERATIONS

As described earlier, caches provide fast temporary data storage, and they make the speedup of memory accesses transparent to the user. In general, the CPU core accesses cache-resident instructions or data through the following procedure:

1. The CPU core, through the on-chip cache controller, attempts to access the next instruction or data in the appropriate cache.
2. The cache controller checks to see if this instruction or data is present in the cache.
   ″ If the instruction/data is present, the CPU core retrieves it. This is called a cache hit.
   ″ If the instruction/data is not present in the cache, the cache controller must retrieve it from memory. This is called a cache miss.
3. The CPU core retrieves the instruction/data from the cache and operation continues.

It is possible for the same data to be in two places simultaneously: main memory and cache. This data is kept consistent through the use of a write-back methodology; that is, modified data is not written back to memory until the cache line is to be replaced.

Instruction and data cache line replacement operations are described in the following sections.

8.3.1 Cache Write Policy

The VR4100 CPU core manages its data cache by using a write-back policy; that is, it stores write data into the cache, instead of writing it directly to memory. Some time later this data is independently written into memory. In the VR4101 implementation, a modified cache line is not written back to memory until the cache line is to be replaced either in the course of satisfying a cache miss, or during the execution of a write-back CACHE instruction.

When the CPU core writes a cache line back to memory, it does not ordinarily retain a copy of the cache line, and the state of the cache line is changed to invalid.

8.4 CACHE STATES

The three terms below are used to describe the state of a cache line:

- Dirty: a cache line containing data that has changed since it was loaded from memory.
- Clean: a cache line that contains data that has not changed since it was loaded from memory.
- Invalid: a cache line that does not contain valid information must be marked invalid, and cannot be used. For example, after a Soft Reset, software sets all cache lines to invalid. A cache line in any other state than invalid is assumed to contain valid information.

The data cache supports three cache states:

- invalid
- valid clean
- valid dirty

The instruction cache supports two cache states:

- invalid
- valid

The state of a valid cache line may be modified when the processor executes a CACHE operation. CACHE operations are described in Chapter 24.

8.5 CACHE STATE TRANSITION DIAGRAMS

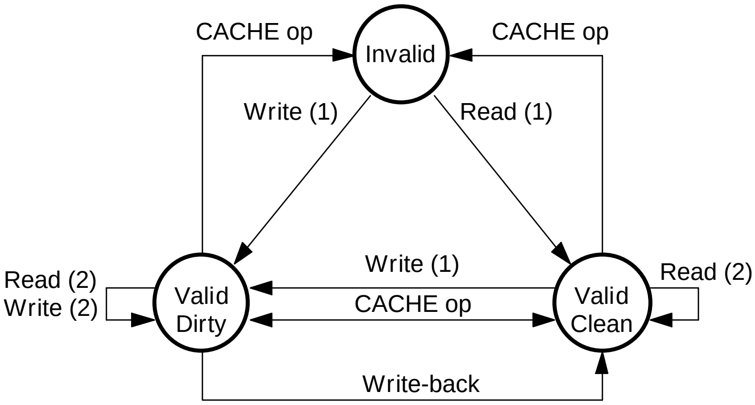
The following section describes the cache state diagrams for the data and instruction caches. These state diagrams do not cover the initial state of the system, since the initial state is system-dependent.

8.5.1 Data Cache State Transition

The following diagram illustrates the data cache state transition sequence. A load or store operation may include one or more of the atomic read and/or write operations shown in the state diagram below, which may cause cache state transitions.

- Read (1) indicates a read operation from memory to cache, inducing a cache state transition.
- Write (1) indicates a write operation from CPU core to cache, inducing a cache state transition.
- Read (2) indicates a read operation from cache to the CPU core, which induces no cache state transition.
- Write (2) indicates a write operation from CPU core to cache, which induces no cache state transition.

Figure 8-6. Data Cache State Diagram

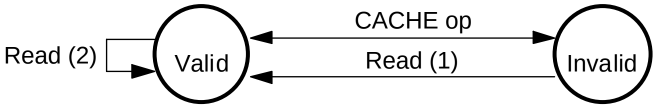


8.5.2 Instruction Cache State Transition

The following diagram illustrates the instruction cache state transition sequence.

- Read (1) indicates a read operation from memory to cache, inducing a cache state transition.
- Read (2) indicates a read operation from cache to the CPU core, which induces no cache state transition.

Figure 8-7. Instruction Cache State Diagram



8.6 CACHE DATA INTEGRITY

The D- and I-cache data RAM arrays are protected by parity. D- and I-cache tag RAM arrays are also protected by parity.

These parity bits are checked for errors on every cache read access. Cache error exception occurs if the CPU core encounters a parity error during an instruction cache access, a data cache access, or memory read access. The CacheErr register indicates the source of the error.

Figure 8-8 to Figure 8-22 shows the parity generation and checking operations for various cache accesses.

Figure 8-8. Data flow on Instruction Fetch

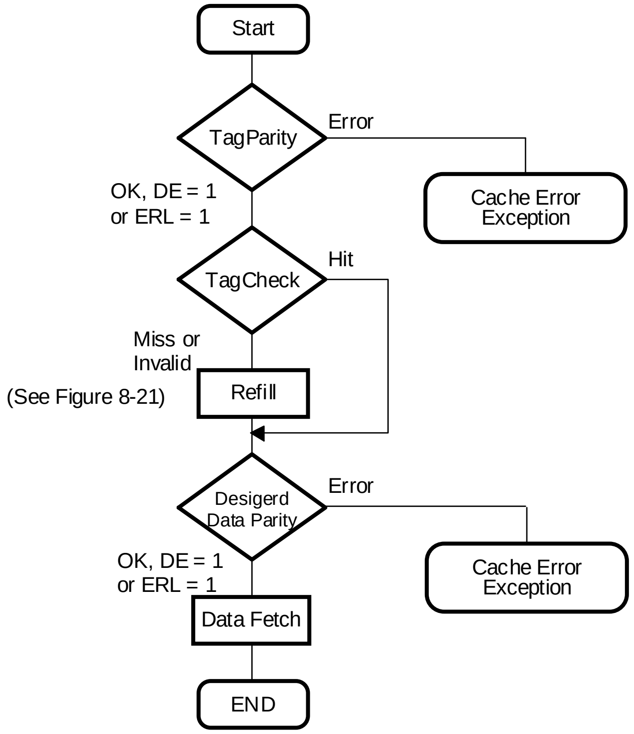


Figure 8-9. Data Integrity on Load Operations

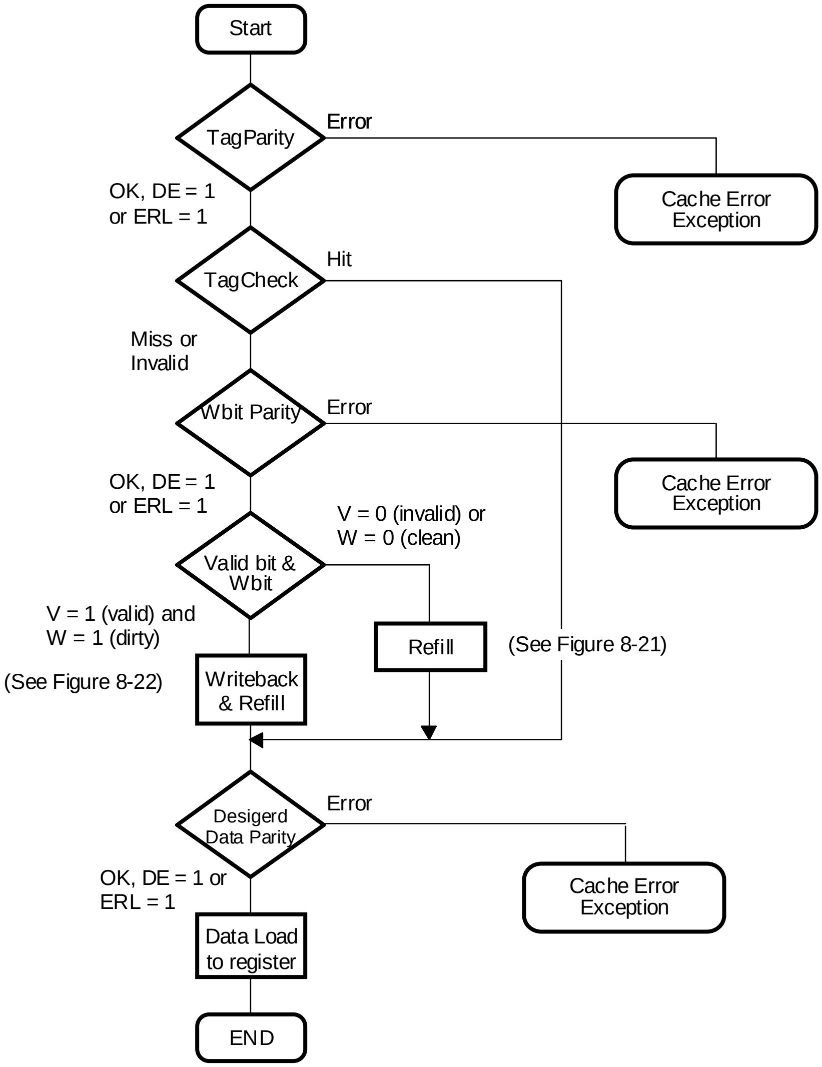


Figure 8-10. Data Integrity on Store Operations

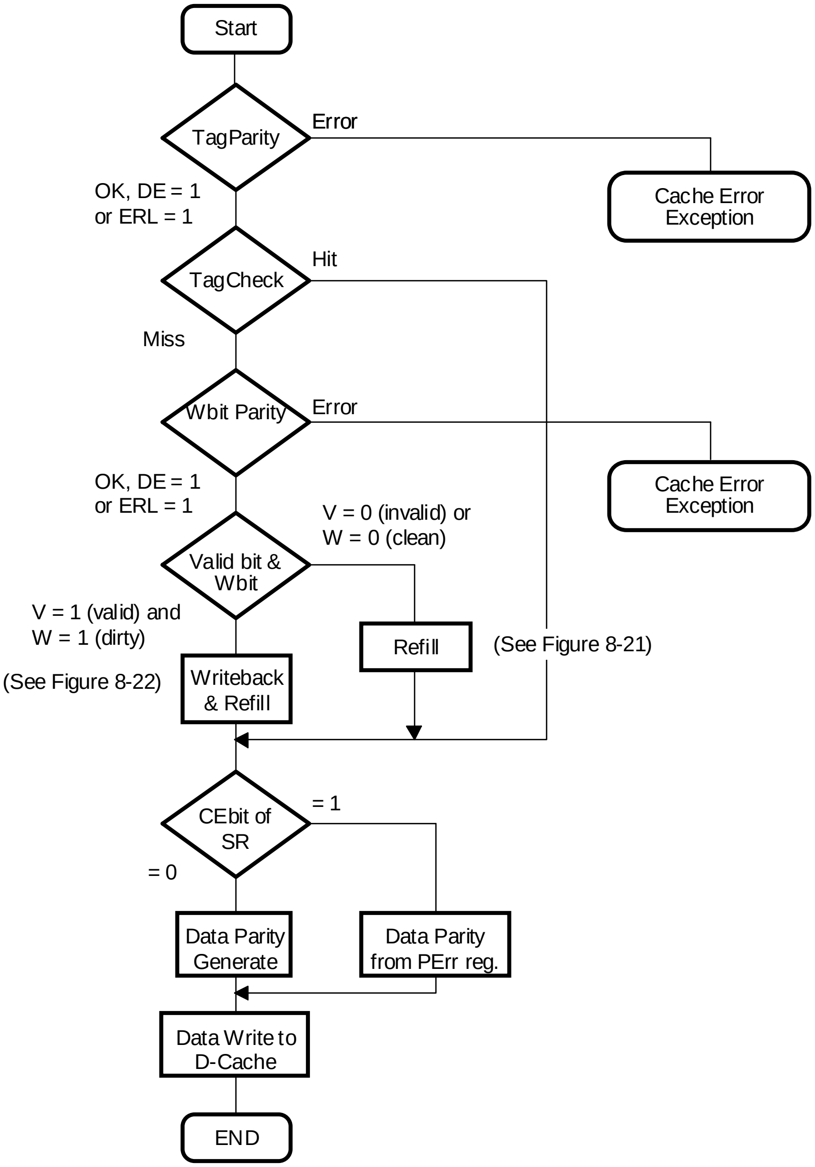


Figure 8-11. Data Integrity on Index_Invalidate Operations

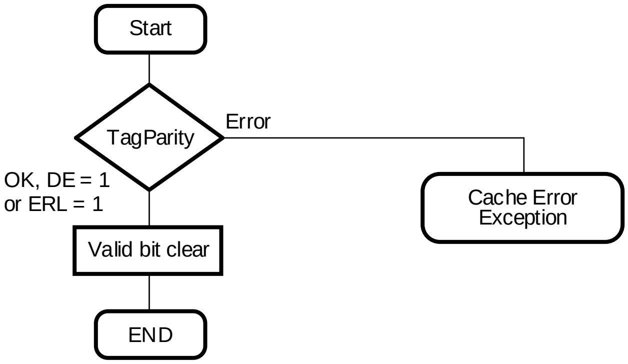


Figure 8-12. Data Integrity on Index_Writeback_Invalidate Operations

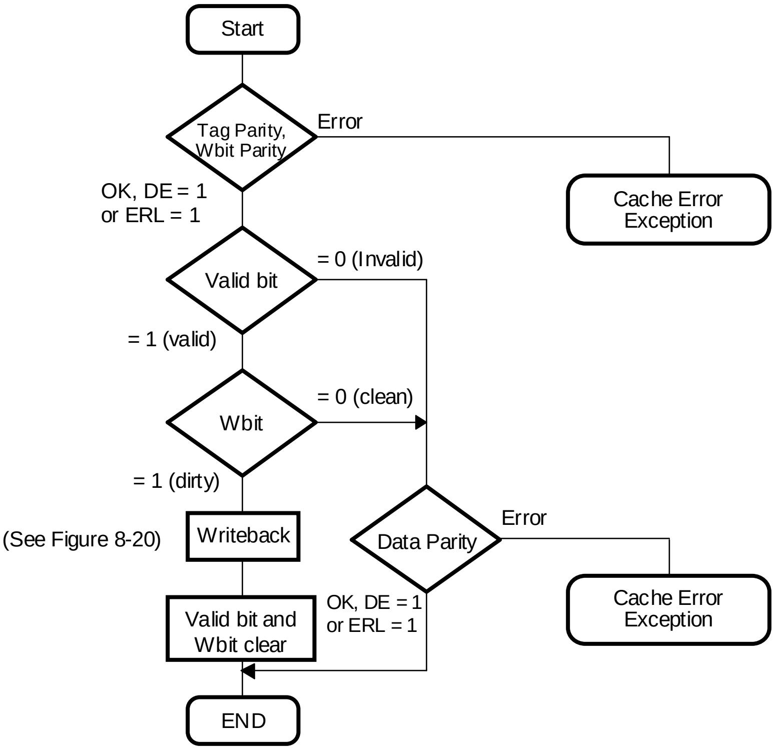


Figure 8-13. Data Integrity on Index_Load_Tag Operations

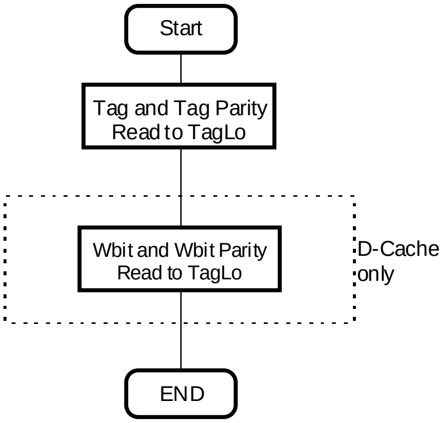

Start
Tag and Tag Parity Read to TagLo
Wbit and Wbit Parity Read to TagLo
D-Cache only
END

Figure 8-14. Data Integrity on Index_Store_Tag Operations

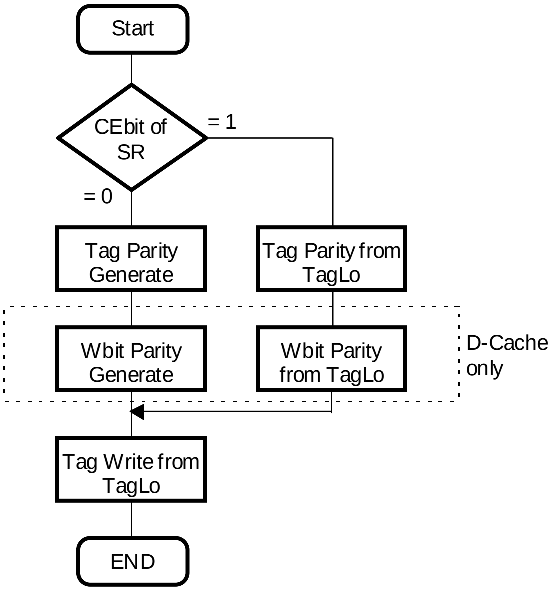


Figure 8-15. Data Integrity on Create_Dirty Operations

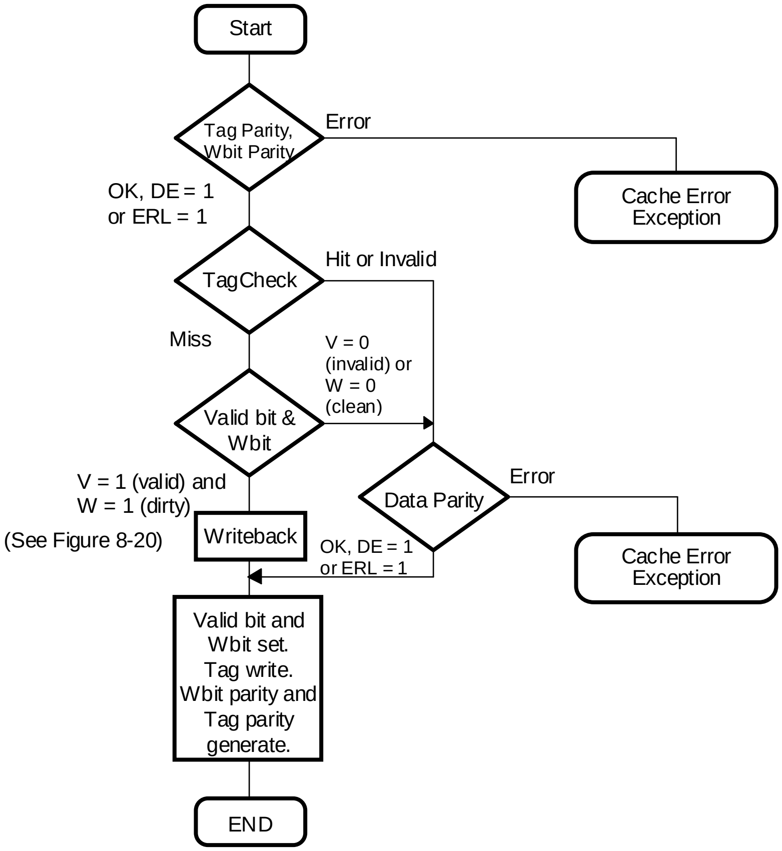


Figure 8-16. Data Integrity on Hit_Invalidate Operations

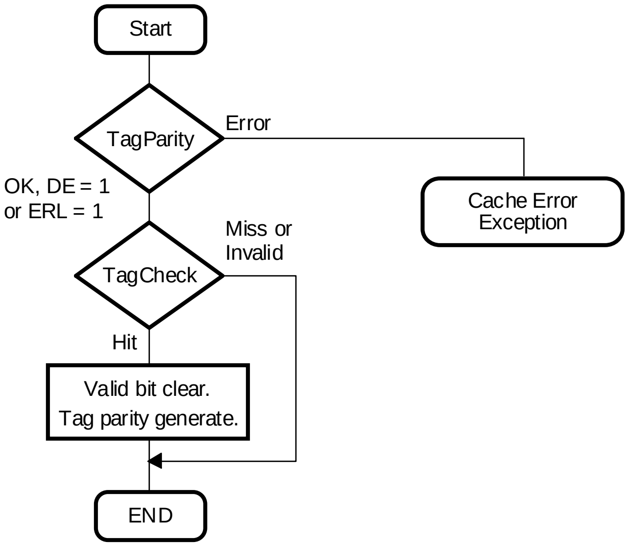


Figure 8-17. Data Integrity on Hit_Writeback_Invalidate Operations

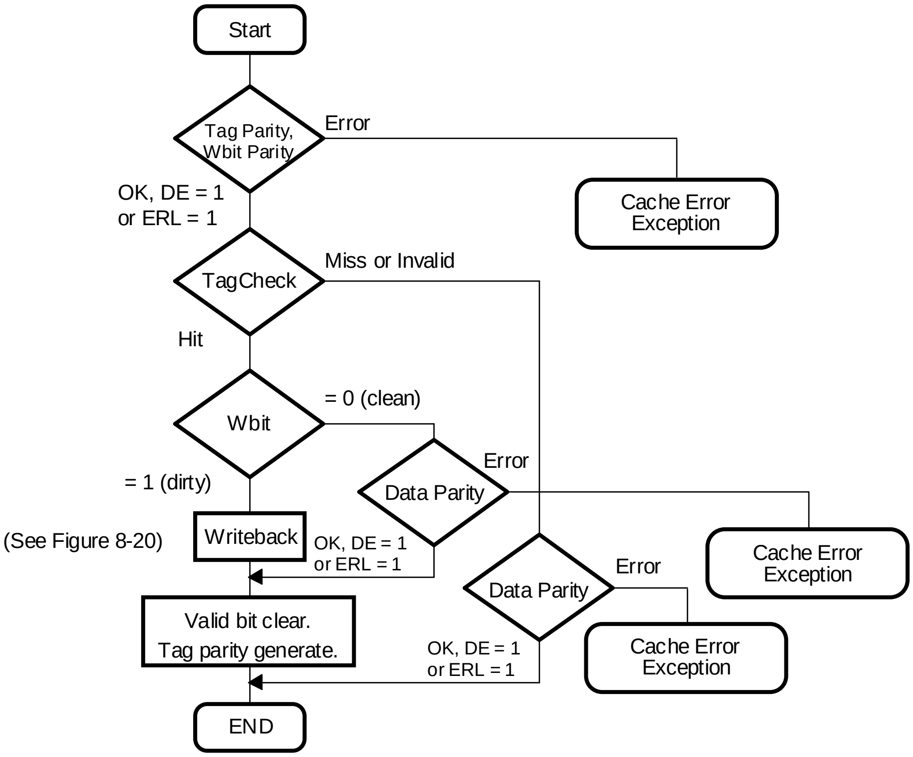


Figure 8-18. Data Integrity on Fill Operations

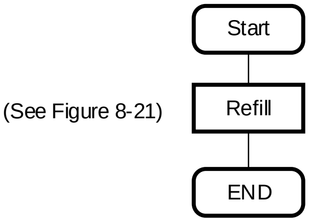


Figure 8-19. Data Integrity on Hit_Writeback Operations

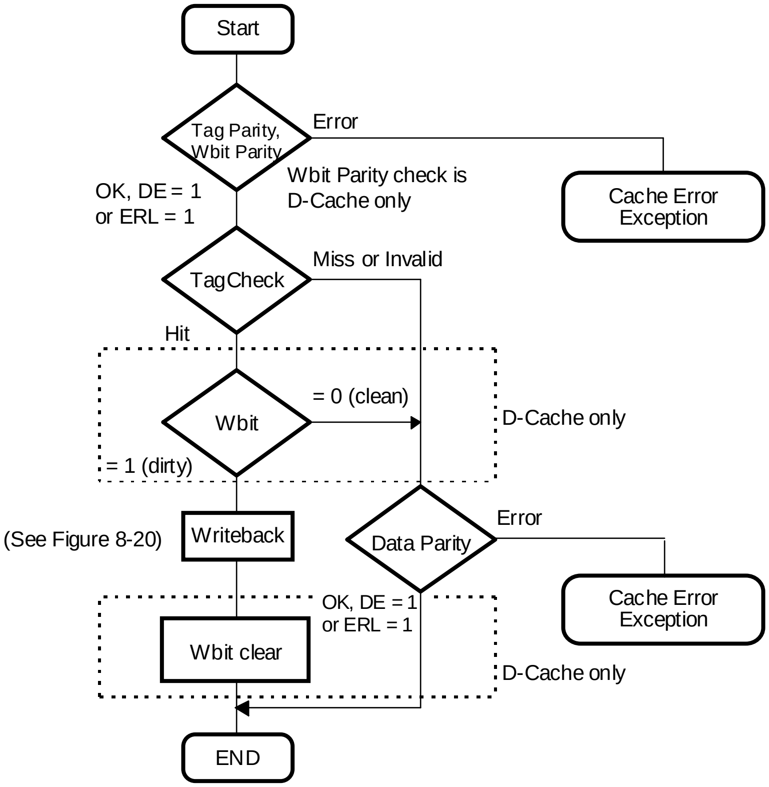


Figure 8-20. Data Integrity on Writeback Flow

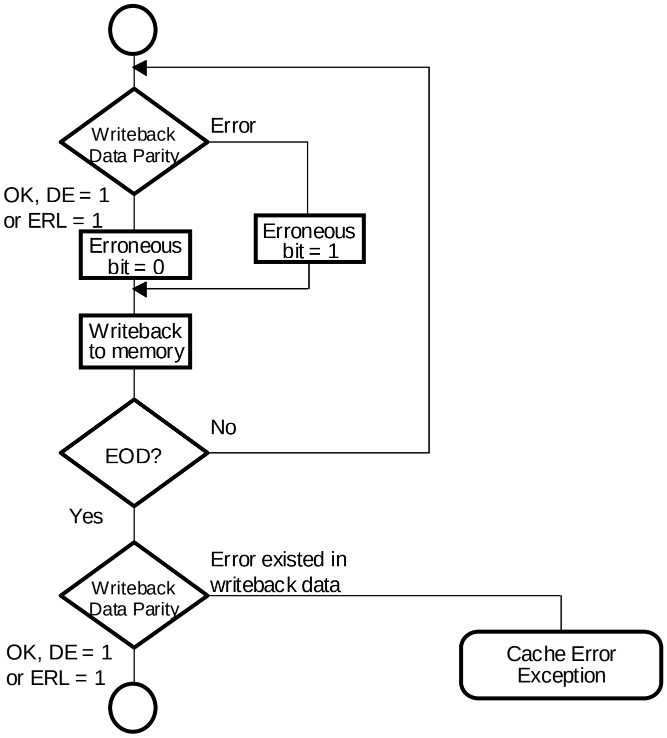


Figure 8-21. Data Integrity on Refill Flow

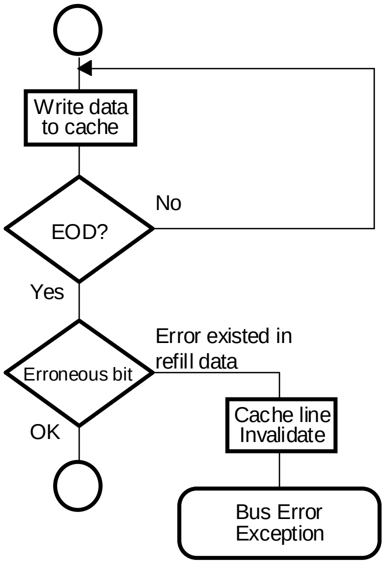
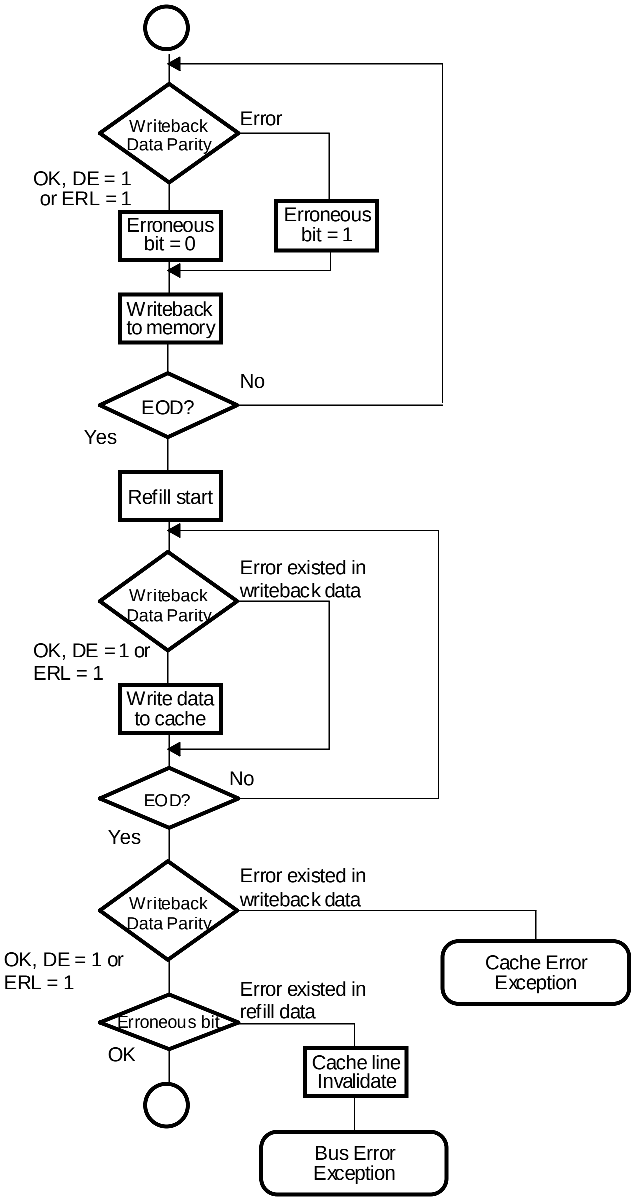


Figure 8-22. Data Integrity on Writeback & Refill Flow



Remark Write-back Procedure:

On a store miss write-back, data and tag parity is checked and data parity is transferred to the write buffer. Byte parity is generated for the physical address and transferred to write buffer. If an error is discovered on the data field, the write back is not terminated; the erroneous data is still written out. If an error is discovered in the tag field, the write-back bus cycle is not issued. In both cases a cache error exception is taken.

If a tag parity error occurs during a CACHE operation, the Cache Error exception is taken and the operation is not permitted to complete.

8.7 MANIPULATION OF THE CACHES BY AN EXTERNAL AGENT

The VR4100 does not provide any mechanisms for an external agent to examine and manipulate the state and contents of the caches.

CHAPTER 9 CPU CORE INTERRUPTS

Four types of interrupt are available on the CPU core. These are:

″ one non-maskable interrupt, NMI

″ five ordinary interrupts

″ two software interrupts

″ one timer interrupt

These are described in this chapter.

9.1 NONMASKABLE INTERRUPT (NMI)

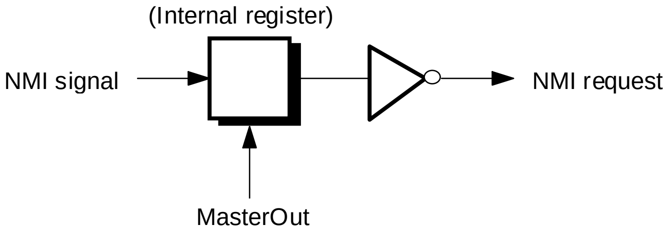
The nonmaskable interrupt is signaled by asserting the NMI signal (internal), forcing the processor to branch to the Reset Exception vector. This signal is latched into an internal register by the rising edge of MasterOut, as shown in Figure 9-1.

NMI only takes effect when the processor pipeline is running.

This interrupt cannot be masked.

Figure 9-1 shows the internal derivation of the NMI signal. The NMI signal is latched into an internal register by the rising edge of MasterOut.

Figure 9-1. Nonmaskable Interrupt Signal



9.2 ORDINARY INTERRUPTS

Ordinary interrupts are set by asserting the Int(4:0) signals (internal). However, Int(4:2) never occur on the VR4101.

These interrupts can be masked with the IM, IE, and EXL fields of the Status register.

9.3 SOFTWARE INTERRUPTS GENERATED IN CPU CORE

Software interrupts generated in the CPU core use bits 1 and 0 of the IP (interrupt pending) field in the Cause register. These may be written by software, but there is no hardware mechanism to set or clear these bits.

After the processing of a software interrupt exception, corresponding bit of the IP field in the Cause register must be cleared before returning to ordinary routine or enabling multiple interrupts.

These interrupts are maskable through the IM, IE, and EXL fields of the Status register.

9.4 TIMER INTERRUPT

The timer interrupt uses bit 15 of the Cause register, which is bit 7 of the IP (interrupt pending) field. This bit is set whenever the value of the Count register equals the value of the Compare register.

This interrupt is maskable through the IM field of the Status register.

9.5 ASSERTING INTERRUPTS

9.5.1 Detecting Hardware Interrupts

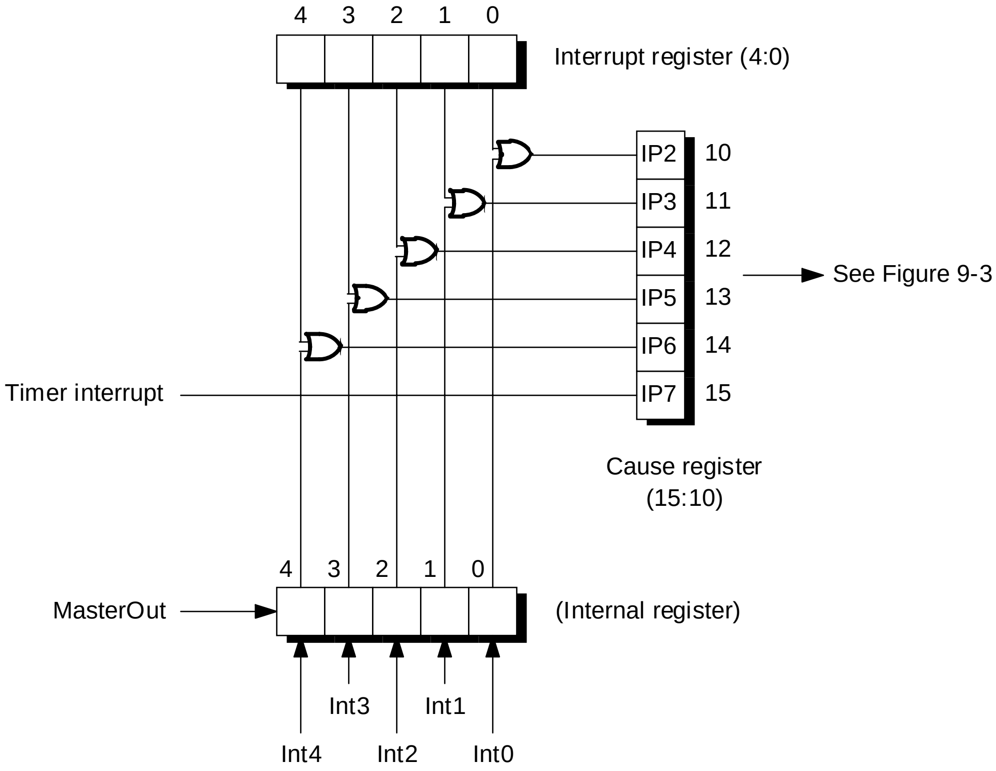
Figure 9-2 shows how the hardware interrupts are readable through the Cause register.

″ The timer interrupt signal, IP7, is directly readable as bit 15 of the Cause register.

″ Bits 4:0 of the Interrupt register are bit-wise ORed with the current value of the Int(4:0) signals and the result is directly readable as bits 14:10 of the Cause register.

IP(1:0*)* of the Cause register, which are described in Chapter 5, are software interrupts. There is no hardware mechanism for setting or clearing the software interrupts.

Figure 9-2. Hardware Interrupt Signals



Remark Int(4:2) never occur in the VR4101.

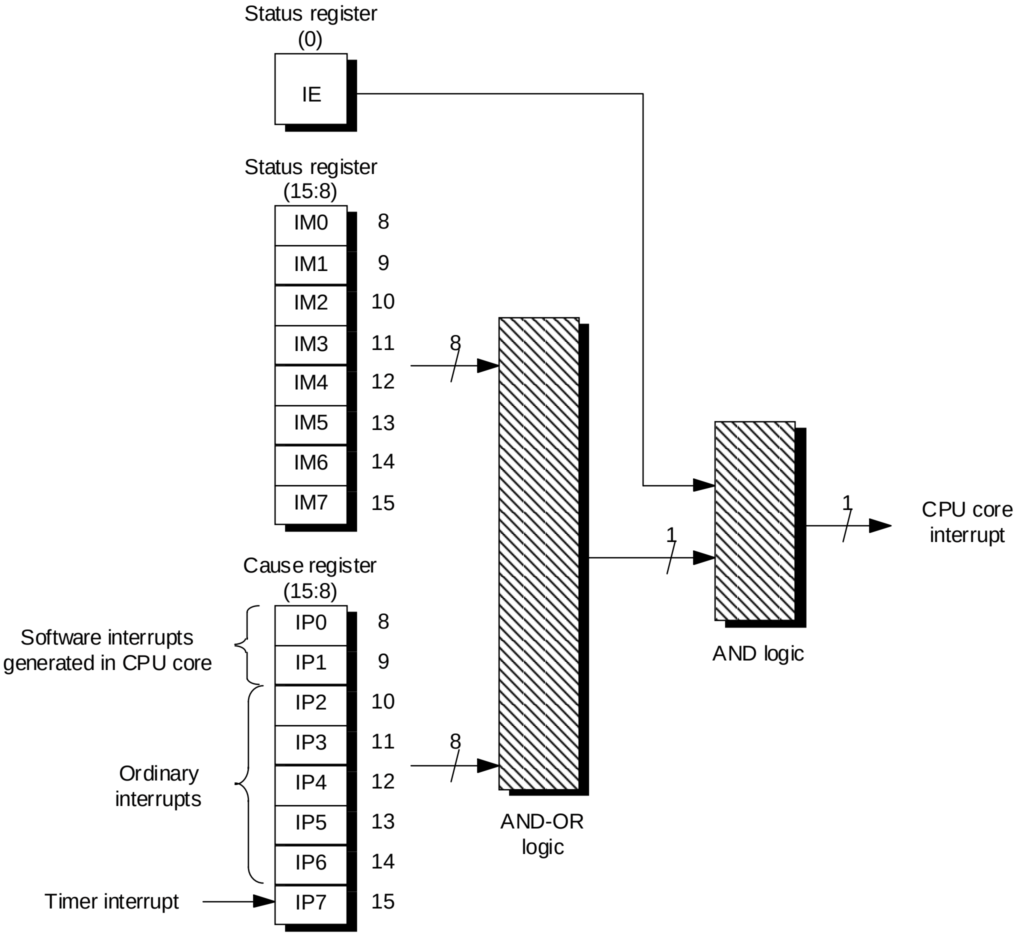
9.5.2 Masking Interrupt Signals

Figure 9-3 shows the masking of the CPU core interrupt signals.

″ Cause register bits 15 to 8 (IP7 to IP0) are AND-ORed with Status register interrupt mask bits 15 to 8 (IM7 to IM0) to mask individual interrupts.

″ Status register bit 0 is a global Interrupt Enable (IE). It is ANDed with the output of the AND-OR logic to produce the CPU core interrupt signal. The EXL bit in the Status register also enables these interrupts.

Figure 9-3. Masking of the CPU Core Interrupts



CHAPTER 10 BCU (BUS CONTROL UNIT)

This chapter explains the operation of the BCU and how to set the registers of the BCU.

10.1 GENERAL

The BCU performs internal data transfer to and from the VR4100 CPU core over the SysAD bus (internal). Externally, it performs data transfer to and from an LCD controller, DRAM, ROM (flash memory or masked ROM), or PCMCIA controller connected to the system bus, via the ADD and DATA buses.

The BCU operates based on TClock, one of internal bus clock.

10.2 REGISTER SET

The following table lists the registers of the BCU.

Table 10-1. BCU Registers

Address	R/W	Register symbols	Function
0x0B00 0000	R/W	BCUCNTREG	BCU Control register
0x0B00 0002	R/W	BCUBRREG	BCU Bus Restrain register
0x0B00 0004	R/W	BCUBRCNTREG	BCU Bus Restrain Count register
0x0B00 0006	R/W	BCUBCLREG	BCU CPU Restrain Disable register
0x0B00 0008	R/W	BCUBCLCNTREG	BCU CPU Restrain Disable Count register
0x0B00 000A	R/W	BCUSPEEDREG	BCU Access Cycle Change register
0x0B00 000C	R/W1C	BCUERRSTREG	BCU Bus Error Status register
0x0B00 000E	R/W	BCURFCNTREG	BCU Refresh Control register
0x0B00 0010	R	PREVIDREG	Peripheral Revision ID register

The function of each of these registers is explained in detail below.

10.2.1 BCUCNTREG

Figure 10-1. BCUCNTREG (0x0B00 0000)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	REF1K	PAGEROM	ROMWEN	SRFSTAT	BCPUREN	Reserved	Reserved	RSTOUT
R/W	R/W	R/W	R/W	R	R/W	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..8]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[7]	REF1K	Sets DRAM refresh interval. 1: 1024 cycles/128 ms 0: 4096 cycles/128 ms
D[6]	PAGEROM	Enables page ROM access. 1: Page ROM bus access 0: Normal ROM bus access
D[5]	ROMWEN	Enables writing of flash memory. 1: Enabled 0: Disabled
D[4]	SRFSTAT	BCU mode (DRAM refresh mode) 1: Self-refresh mode 0: CBR refresh mode
D[3]	BCPUREN	CPU bus cycle control enable bit 1: Enables CPU bus control 0: Disables CPU bus control
D[2..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	RSTOUT	RSTOUT control bit 1: Sets the RSTOUT pin to High level 0: Clears the RSTOUT pin to Low level

This register sets parameters such as the bus cycle of the bus interface.

The settings of the BCUBRREG is effective when the BCPUREN bit is 1 (see 10.2.2).

10.2.2 BCUBRREG

Figure 10-2. BCUBRREG (0x0B00 0002)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	BCPUT [15]	BCPUT [14]	BCPUT [13]	BCPUT [12]	BCPUT [11]	BCPUT [10]	BCPUT [9]	BCPUT [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	BCPUT [7]	BCPUT [6]	BCPUT [5]	BCPUT [4]	BCPUT [3]	BCPUT [2]	BCPUT [1]	BCPUT [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	BCPUT[15..0]	Sets BCU transaction interval. BCU transaction interval = BCPUT[15..0] * TClock period

This register is used to set the interval applied to transactions performed between the BCU and CPU core.

When the BCPUREN bit of the BCUCNTREG is set to 1, the value set by this register is used as the BCU transaction interval (see 10.2.1).

10.2.3 BCUBRCNTREG

Figure 10-3. BCUBRCNTREG (0x0B00 0004)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	BTCNT [15]	BTCNT [14]	BTCNT [13]	BTCNT [12]	BTCNT [11]	BTCNT [10]	BTCNT [9]	BTCNT [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	BTCNT [7]	BTCNT [6]	BTCNT [5]	BTCNT [4]	BTCNT [3]	BTCNT [2]	BTCNT [1]	BTCNT [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	BTCNT[15..0]	Counts BCU transactions. Counts current BCU transactions.

This register is used to read or write the BCU transaction count.

The value of BTCNT is incremented in synchronization with TClock. When the BCPUREN bit of BCUCNTREG is set to 1, and provided the count of this register is the same as the value set with BCUBRREG, the contents of this register are cleared to 0.

10.2.4 BCUBCLREG

Figure 10-4. BCUBCLREG (0x0B00 0006)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	BCLR [15]	BCLR [14]	BCLR [13]	BCLR [12]	BCLR [11]	BCLR [10]	BCLR [9]	BCLR [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	BCLR [7]	BCLR [6]	BCLR [5]	BCLR [4]	BCLR [3]	BCLR [2]	BCLR [1]	BCLR [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	BCLR[15..0]	Number of repetitions required to enable BCU transaction interval.

This register is used to set the number of repetitions required to enable the BCU transaction interval set with BCUBRREG.

When the BCU transaction has been performed the number of times set with this register, the BCPUREN bit of BCUCNTREG is cleared to 0.

10.2.5 BCUBCLCNTREG

Figure 10-5. BCUBCLCNTREG (0x0B00 0008)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	BCPUC [15]	BCPUC [14]	BCPUC [13]	BCPUC [12]	BCPUC [11]	BCPUC [10]	BCPUC [9]	BCPUC [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	BCPUC [7]	BCPUC [6]	BCPUC [5]	BCPUC [4]	BCPUC [3]	BCPUC [2]	BCPUC [1]	BCPUC [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	BCPUC[15..0]	Counts the number of times the BCU transaction is performed. Number of times BCU transaction has been performed.

This register is used to count the number of times the BCU transaction, set with BCUBRREG, has been performed.

The number of times the current BCU transaction has been performed can be both read and written. The value of BCPUC is incremented each time the BCU transaction is performed. While the BCPUREN bit of BCUCNTREG is set to 1, and provided the value of this register is the same as the value set with BCUBCLREG, the contents of this register are cleared to 0.

10.2.6 BCUSPEEDREG

Figure 10-6. BCUSPEEDREG (0x0B00 000A) (1/2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	WPROM [1]	WPROM [0]	Reserved	Reserved	WLCDA[1]	WLCDA[0]
R/W	R	R	R/W	R/W	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	WISAA[2]	WISAA[1]	WISAA[0]	Reserved	WROMA[2]	WROMA[1]	WROMA[0]
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	1	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13..12]	WPROM[1..0]	Page ROM access speed (Tprom) 11: Reserved for future use. 10: 1 TClock 01: 2 TClocks 00: 3 TClocks (initial value)
D[11..10]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[9..8]	WLCDA[1..0]	LCD access speed (Tlcd) 11: 2 TClocks 10: 4 TClocks 01: 6 TClocks 00: 8 TClocks (initial value)
D[7]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.

Figure 10-6. BCUSPEEDREG (0x0B00 000A) (2/2)

Bit position	Bit name	Function
D[6..4]	WISAA[2..0]	ISA access speed (Tisa) 111: Reserved for future use. Operation is not guaranteed if this value is set. 110: Reserved for future use. Operation is not guaranteed if this value is set. 101: 3 TClocks 100: 4 TClocks 011: 5 TClocks 010: 6 TClocks 001: 7 TClocks (initial value) 000: 8 TClocks
D[3]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[2..0]	WROMA[2..0]	ROM access speed (Trom) 111: 2 TClocks 110: 3 TClocks 101: 4 TClocks 100: 5 TClocks 011: 6 TClocks 010: 7 TClocks 001: 8 TClocks 000: 9 TClocks (initial value)

This register sets the access speeds of LCD, ISA, page ROM, and ROM.

When the WLCDA[1..0], WPROM[1..0], WISAA[2..0], and WROMA[2..0] bits are set to 0, the lowest speed is set. When these bits are set to 1, the highest speed is set.

The value of WPROM[1..0] is effective only when the PAGEROM bit of BCUCNTREG is set to 1.

10.2.7 BCUERRSTREG

Figure 10-7. BCUERRSTREG (0x0B00 000C)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	BERRST
R/W	R	R	R	R	R	R	R	R/W1C
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	BERRST	Bus error status 1: Bus error 0: Normal

This register indicates the occurrence of a bus error interrupt.

By setting the BERRST bit to 1, the bus error interrupt is cleared.

10.2.8 BCURFCNTREG

Figure 10-8. BCURFCNTREG (0x0B00 000E)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	BRF [12]	BRF [11]	BRF [10]	BRF [9]	BRF [8]
R/W	R	R	R	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	BRF [7]	BRF [6]	BRF [5]	BRF [4]	BRF [3]	BRF [2]	BRF [1]	BRF [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..13]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[12..0]	BRF[12..0]	Refresh cycle counter

This register indicates the current count of the refresh cycle.

10.2.9 PREVIDREG

Figure 10-9. PREVIDREG (0x0B00 0010)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserve d	Reserve d	Reserve d	Reserve d	MJREV [3]	MJREV [2]	MJREV [1]	MJREV [0]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserve d	Reserve d	Reserve d	Reserve d	MNRE V [3]	MNREV [2]	MNRE V [1]	MNREV [0]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	x	x	x	x

Bit position	Bit name	Function
D[15..12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..8]	MJREV[3..0]	Major revision number
D[7..4]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[3..0]	MNREV[3..0]	Minor revision number

Remark x: undefined

This register indicates the revision of the peripheral units of the VR4101.

The revision number is stored as a value in the form *y.x*, where *y* is a major revision number and *x* is a minor revision number.

Major revision number and minor revision number can distinguish the revision of the peripheral units, however there is no guarantee that changes to the peripheral units will necessarily be reflected in this register, or that changes to the revision number necessarily reflect real units' changes. For this reason, these values are not listed and software should not rely on the revision number in PREVIDREG to characterize the units.

10.3 MEMORY ACCESS BY BCU

The Bus Control Unit (BCU) is the unit to perform the initiation of bus cycles and bus arbitration for the CPU core or DMAC to access an external device or a built-in I/Os.

10.3.1 Address Map

The address map accessible through the BCU is shown below.

Table 10-2. Address Map of the VR4101

Physical address	Space
0x1FFF FFFF - 0x1F00 0000 0x18FF FFFF - 0x1800 0000	ROM
0x17FF FFFF - 0x1600 0000	Expansion I/O
0x15FF FFFF - 0x1400 0000	Expansion Memory
0x0BFF FFFF - 0x0B00 0000	Register
0x0AFF FFFF - 0x0A00 0000	VRAM (LCD)
0x03FF FFFF - 0x0000 0000	DRAM
0x1EFF FFFF - 0x1900 0000 0x13FF FFFF - 0x0C00 0000 0x09FF FFFF - 0x0400 0000	Address space reserved for the future

10.3.2 Address Space for ROM

Address space for the ROM is selected by the ROM chip select terminals as below.

Table 10-3. Detailed Address Map for the ROM

Physical address	ROM chip select terminal
0x1FFF FFFF - 0x1FC0 0000, 0x18FF FFFF - 0x18C0 0000	ROMCS*[3]
0x1FBF FFFF - 0x1F80 0000, 0x18BF FFFF - 0x1880 0000	ROMCS*[2]
0x1F7F FFFF - 0x1F40 0000, 0x187F FFFF - 0x1840 0000	ROMCS*[1]
0x1F3F FFFF - 0x1F00 0000, 0x183F FFFF - 0x1800 0000	ROMCS*[0]

10.3.3 Address Space for Expansion Bus

Expansion bus has two access types, I/O access and memory access, and each has two modes, 16-bit device mode and 8-bit device mode, which are automatically selected by physical address output by CPU core. Address space for each mode is selected by the read/write terminals for I/O access or memory access as below.

(1) Expansion I/O access

Table 10-4. 16-Bit Device Mode for the Expansion I/O

Physical address	Read/write terminal for I/O access
0x17FF FFFF - 0x1720 0000	RFU
0x171F FFFF - 0x1700 0000	IOR* / IOW*

Table 10-5. 8-Bit Device Mode for the Expansion I/O

Physical address	Read/write terminal for I/O access
0x16FF FFFF - 0x1620 0000	RFU
0x161F FFFF - 0x1600 0000	IOR* / IOW*

(2) Expansion memory access

Table 10-6. 16-Bit Device Mode for the Expansion Memory

Physical address	Read/write terminal for memory access
0x15FF FFFF - 0x1520 0000	RFU
0x151F FFFF - 0x1500 0000	MEMR* / MEMW*

Table 10-7. 8-Bit Device Mode for the Expansion Memory

Physical address	Read/write terminal for memory access
0x14FF FFFF - 0x1420 0000	RFU
0x141F FFFF - 0x1400 0000	MEMR* / MEMW*

10.3.4 Address Space for Registers

Address space for the registers which belong to on-chip peripheral units is mapped at every unit as below. Refer to chapters of each unit for detailed address of each register.

Table 10-8. Register Address Space for Peripheral Units

Physical address	Unit
0x0BFF FFFF - 0x0B00 01C0	RFU
0x0B00 01BF - 0x0B00 01A0	DSIU
0x0B00 019F - 0x0B00 0180	KIU
0x0B00 017F - 0x0B00 0160	ADU
0x0B00 015F - 0x0B00 0140	SIU
0x0B00 013F - 0x0B00 0120	PIU
0x0B00 011F - 0x0B00 0100	GIU
0x0B00 00FF - 0x0B00 00E0	DSU
0x0B00 00DF - 0x0B00 00C0	RTC
0x0B00 00BF - 0x0B00 00A0	PMU
0x0B00 009F - 0x0B00 0080	ICU
0x0B00 007F - 0x0B00 0060	CMU
0x0B00 005F - 0x0B00 0040	DCU
0x0B00 003F - 0x0B00 0020	DMAA
0x0B00 001F - 0x0B00 0000	BCU

10.3.5 Address Space for LCD

Address space at LCD access is selected by LCD controller chip select terminal as below.

Table 10-9. Detailed LCD Address Space

Physical address	LCD controller chip select terminal
0x0AFF FFFF - 0x0A20 0000	RFU
0x0A1F FFFF - 0x0A00 0000	LCDCS*

10.3.6 Address Space for DRAM

Address space at DRAM access is selected by RAS terminals for DRAM as below.

Table 10-10. Detailed DRAM Address Space

Physical address	RAS terminals for DRAM
0x03FF FFFF - 0x0080 0000	RFU
0x007F FFFF - 0x0060 0000	MRAS*[3]
0x005F FFFF - 0x0040 0000	MRAS*[2]
0x003F FFFF - 0x0020 0000	MRAS*[1]
0x001F FFFF - 0x0000 0000	MRAS*[0]

10.4 CONNECTION OF ADDRESS TERMINALS

Physical address output from CPU core is provided to external devices through ADD bus. The correspondence between the address output to ADD bus and the address bits of external devices is different from the external devices as shown in Table 10-11. Therefore, connect ADD bus and address bits of the external device as shown in Table 10-12.

Table 10-11. Address Bit Correspondence between ADD Bus and External Devices

Devices connected	ADD bus																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ROM	21	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Expansion bus LCD DRAM (row)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
DRAM (column)	0	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	19	20	19	20

Table 10-12. Address Connection Table with External Devices

VR4101 terminal	Address terminals of external devices		
	ROM	Expansion bus VRAM (LCDC)	DRAM
ADD[0]	20	0	—
ADD[1]	0	1	—
ADD[2]	1	2	—
ADD[3]	2	3	—
ADD[4]	3	4	—
ADD[5]	4	5	—
ADD[6]	5	6	—
ADD[7]	6	7	—
ADD[8]	7	8	—
ADD[9]	8	9	0
ADD[10]	9	10	1
ADD[11]	10	11	2
ADD[12]	11	12	3
ADD[13]	12	13	4
ADD[14]	13	14	5
ADD[15]	14	15	6
ADD[16]	15	16	7
ADD[17]	16	17	8
ADD[18]	17	18	9
ADD[19]	18	19	10
ADD[20]	19	20	11

10.5 NOTES FOR USING BCU

10.5.1 CPU Core Bus Modes

The VR4101 is designed on the proposition that the CPU core is set the mode for bus interface as below:

Writeback data rate: DxDx
Accelerate data mode: R4x00 compatible mode

Therefore, set the Config Register as below:

EP field: 0011
AD bit: 0

10.5.2 Access Data Size

The VR4101 has a restricted access size for each address space.

The access size for each address space is show below.

Table 10-13. Access Size for Each Address Space

Address space	R/W	Access size (byte)						Remarks
		16	8	4	3	2	1	
ROM	R	A	A	A	A	A	A	
Flash memory	W	N/A	N/A	N/A	N/A	A	N/A	
Expansion bus 8-bit device mode	R/W	A	A	A	N/A	A	*	
Expansion bus 16-bit device mode	R/W	A	A	A	N/A	A	**	
Built-in I/O resource (register)	R/W	N/A	N/A	A	N/A	A	N/A	
LCD controller	R/W	N/A	A	A	N/A	A	A	
DRAM	R/W	A	A	A	A	A	A	Use this with non-cache

* When performing 1-byte access in the expansion bus 8-bit device mode, access is made using only DATA[7..0] of DATA[15..0].

** When performing 1-byte access in the expansion bus 16-bit device mode, DATA[7..0] is used for the access to even-numbered addresses, and DATA[15..8] for the access to odd-numbered addresses.

10.5.3 ROM Interface

(1) ROM/Page-ROM/Flash Memory switching

The VR4101 performs Ordinary ROM/Page-ROM/Flash Memory mode switching by setting of the ROMWEN bit and PAGEROM bit of the BCUCNTREG. In Ordinary ROM mode or Flash Memory mode, the VR4101 can access to memories regardless of its mode name. Table 10-14 shows accessible memory types and methods of access in each mode.

Table 10-14. Summary of ROM Modes

Mode	Setting		Accessible device		
	ROMWEN	PAGEROM	Memory read	Flash memory register read	Flash memory write
Ordinary ROM	0	0	Ordinary ROM Page-ROM Flash memory	N/A	N/A
Page-ROM	0	1	Page-ROM	N/A	N/A
Flash Memory	1	x	Ordinary ROM Page-ROM Flash memory	Flash memory	Flash memory

Note The default setting is the Ordinary ROM mode.

x: don’t care

(2) Setting of access speed

The VR4101 can change the access speed during operation in the Ordinary ROM mode or Page-ROM mode. Refer to 10.6.1 for details.

10.5.4 Flash Memory Interface

(1) Restrictions on each mode

Flash memory interface has two mode as follows:

Ordinary ROM mode (exclusively for memory read)
Flash Memory mode (for write and register read)

Restrictions in each mode are as described below.

(a) Restrictions in the Ordinary ROM mode

Write is prohibited.

Even if write is performed, the LCDCS* (ROMWE*) terminal is not asserted.

Flash memory register read is prohibited.

The Ordinary ROM mode is the mode to issue the bus cycle suitable for memory read. Because the Flash memory uses different AC characteristics for register mode and memory mode, correct data cannot be obtained if read of Flash memory register is performed in this mode.

(b) Restrictions in the Flash memory mode

When performing write to the Flash memory, be sure to access with two bytes.

(2) Example of write sequence to Flash memory

Example of write sequence to Flash memory is shown below.

Caution Confirmation of the operation of this example on the actual system is not yet performed.

1. Using GPIO as the output port, apply write voltage (V_{PP}) to Flash memory.

   If the built-in GPIO of the VR4101 is not available, install an output port on the outside and control the write voltage.
2. Set the VR4101 to the Flash memory mode (Set the ROMWEN bit of BCUCNTREG to 1).
3. Wait until the write voltage to Flash memory becomes stable.
4. Issue the write command to Flash memory from the VR4101.
5. Write data to Flash memory from the VR4101.
6. Wait until the write completion signal of Flash memory (ry/by) becomes stable.
7. Wait until the write completion signal of Flash memory notice the completion of write.

   Completion of the write to Flash memory can be known by the interruption with the Flash memory write completion signal (ry/by) or palling the Flash memory register.
8. Read the Flash memory register.

   If the write has succeeded, perform processing from "9."

   If the write has failed, perform processing from "12."
9. When writing new data to Flash memory, perform processing from "4."

When finishing the read to Flash memory, perform processing from "10."

10. Compare the data written to Flash memory with the original data.

    If these data accord, perform processing of "11."

    If these data do not accord,

    When performing write again, perform processing from "1."

    When ending the process, perform processing from "11."

11. Drop the write voltage of Flash memory (V_{PP}), release the Flash memory mode, and end the processing.

12. Clear the error information from the Flash memory register.

    When performing write again

    If the write voltage was too low, perform processing from "1."

    In other cases, perform processing from "4."

    When ending the process, perform processing of "11."

10.5.5 Expansion Bus Interface

Because the VR4101 does not support dynamic bus sizing, it is specified that only an 8-bit access is allowed as an access to an 8-bit device.

Dynamic bus sizing is the function to change the DATA bus width dynamically in response to the sizing request from the target device (e.g. the bus sizing using the MEMCS16 and -IOCS16 signals of the ISA bus).

(1) Access size in each mode

Restrictions on the access to each of an 8-bit device and 16-bit device are as described below.

(a) 8-bit device mode

Table 10-15. Restrictions on the Access to an 8-bit Device in the 8-bit Device Mode

Access size	Read	Write
Odd-numbered bytes	A	A
Even-numbered bytes	A	A
2 bytes	N/A	N/A
4 bytes	N/A	N/A
8 bytes	N/A	N/A
16 bytes	N/A	N/A

Table 10-16. Restrictions on the Access to a 16-bit Device in the 8-bit Device Mode

Access size	Read	Write
Odd-numbered bytes	N/A	N/A
Even-numbered bytes	A	A
2 bytes	A	A
4 bytes	A	A
8 bytes	A	A
16 bytes	A	A

(b) 16-bit device mode

Table 10-17. Restrictions on the Access to an 8-bit Device in the 16-bit Device Mode

Access size	Read	Write
Odd-numbered bytes	N/A	N/A
Even-numbered bytes	A	A
2 bytes	N/A	N/A
4 bytes	N/A	N/A
8 bytes	N/A	N/A
16 bytes	N/A	N/A

Table 10-18. Restrictions on the Access to a 16-bit Device in the 16-bit Device Mode

Access size	Read	Write
Odd-numbered bytes	A	A
Even-numbered bytes	A	A
2 bytes	A	A
4 bytes	A	A
8 bytes	A	A
16 bytes	A	A

(2) Mode switching

Switching between the 8-bit device mode and 16-bit device mode is effected by the physical address output from the CPU core. Refer to 10.3.3 for details.

10.5.6 LCD Controller Interface

(1) Access size

Be sure to perform the access on the LCD controller interface with 1 byte, 2 bytes, 4 bytes, or 8 bytes.

(2) Reversal of data

The VR4101 reverses the data read and written from and to the LCD controller interface in terms of bits.

Table 10-19. Example of Reversal in Terms of Bits of the Internal Data of the VR4101 and the Data on the DATA[15.0] Terminal

Internal Data of the VR4101	Data on the DATA[15.0] Terminal
0x0000	0xFFFF
0xA5A5	0x5A5A
0x1234	0xEDCB

10.5.7 Notice of an Illegal Access

(1) Types of illegal access

The VR4101 notices the occurrence of an illegal access to the CPU core.

Dead rock of the bus

Because no ready signal is returned from the expansion bus or LCD controller interface when two or more CBR refreshes are disabled, it is judged as of a dead lock of the bus and the occurrence of an illegal access is noticed.

Address space reserved for the future

When the processor has accessed to the following address, the occurrence of an illegal access is noticed.

Access to 0x1EFF FFFF - 0x1900 0000
Access to 0x13FF FFFF - 0x0C00 0000
Access to 0x09FF FFFF - 0x0400 0000

(2) Methods for noticing an illegal access

Methods for noticing to the CPU core are as follows:

Table 10-20. Methods for Noticing an Illegal Access

Type of access	Method for noticing an illegal access
Processor read request	Noticed by a bus error indication on the SysCmd bus
Processor write request	Noticed as an interrupt exception (Int0)

Note The clearance of the interrupt factor by a processor write request is effected by writing 1 to bit 1 of the BCUERRSTREG.

10.6 BUS OPERATION

The BCU operates based on TClock, one of internal bus clock.

10.6.1 ROM Access

The VR4101 supports the following three modes for ROM access.
Mode setting is effected by the PAGEROM bit and ROMWEN bit of BCUCNTREG.

Ordinary ROM read mode (ROMWEN, PAGEROM = 00)

Page-ROM read mode (ROMWEN, PAGEROM = 01)

Flash Memory mode (ROMWEN = 1)

(1) Ordinary ROM read mode

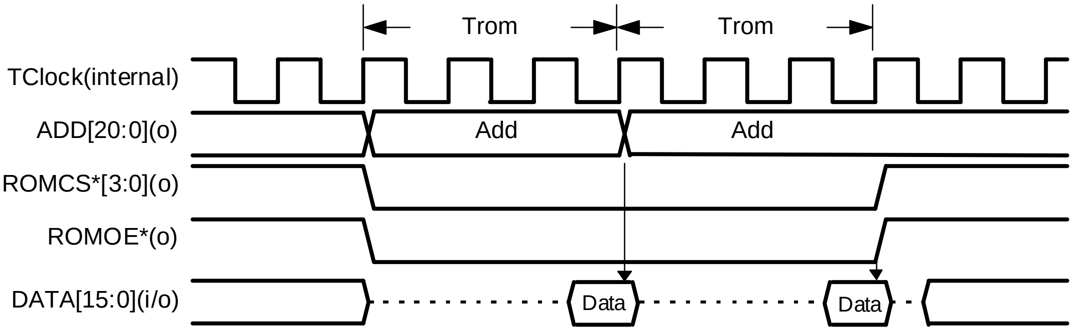
Set to ROMWEN = 0, PAGEROM =0.

The access time can be set by WROMA[2:0] (BCUSPEEDREG[2:0]).

Table 10-21. Access Time in the Ordinary ROM Read Mode

WROMA[2:0]	Trom(TClock)
0 0 0	9
0 0 1	8
0 1 0	7
0 1 1	6
1 0 0	5
1 0 1	4
1 1 0	3
1 1 1	2

Figure 10-10. ROM 4-Byte Read (WROMA[2:0] = 110)



Remark Broken lines indicate high impedance.

DATA is sampled at the rising edge of TClock following the last TClock of the Trom state.

Types of bus operation of the ordinary ROM are as described below.

1-byte read, 2-byte read, 3-byte read, 1-word read (4-byte), 2-word read, 4-word read

(2) Page-ROM read mode

Set to ROMWEN = 0, PAGEROM = 1.

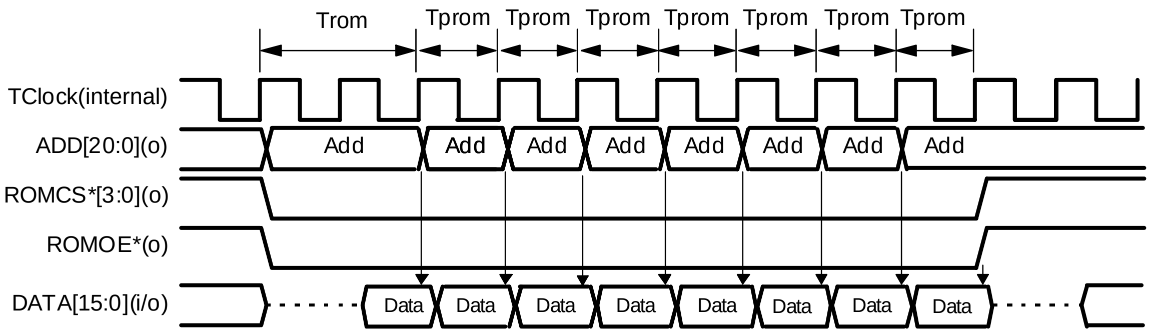
The access time can be set by WROMA[2:0] (BCUSPEEDREG[2:0]), WPROM[1:0] (BCUSPEEDREG[13:12]).

Table 10-22. Access Time in the Page-ROM Read Mode

WROMA[2:0]	Trom(TClock)
0 0 0	9
0 0 1	8
0 1 0	7
0 1 1	6
1 0 0	5
1 0 1	4
1 1 0	3
1 1 1	2

WPROM[1:0]	Tprom(TClock)
0 0	3
0 1	2
1 0	1
1 1	RFU

Figure 10-11. Page-ROM 4-Byte Read (WROMA[2:0] = 110, WPROM[1:0] = 01)



Remark Broken lines indicate high impedance.

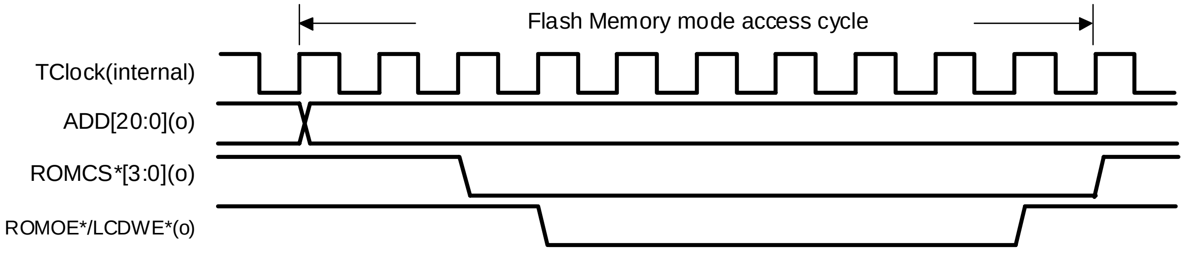
(3) Flash Memory mode

Set to ROMWEN = 1.

This is the mode to satisfy the sequence of the write to the Flash memory and access to the Flash memory register. Read from the Flash memory is also available in this mode.

The access time is constant in this mode.

Figure 10-12. Flash Memory 2-byte Access



10.6.2 Expansion Bus Interface

Bus specification for the 16-bit device mode and 8-bit device mode are as describe below.

Table 10-23. Bus Specifications for the 8-Bit Device Mode

SHB	ADD[0]	Access size	Write		Read		Remarks
			DATA[15:8]	DATA[7:0]	DATA[15:8]	DATA[7:0]	
0	0	2 byte	A	A	*	*	
0	1	1 byte	N/A	N/A	---	---	Note 1
1	0	1 byte	N/A	A	---	*	Byte access to even-numbered address
1	1	1 byte	A	DATA[15:8] Copy	---	*	Byte access to odd-numbered address

Table 10-25. Bus Specifications for the 16-Bit Device Mode

SHB	ADD[0]	Access size	Write		Read		Remarks
			DATA[15:8]	DATA[7:0]	DATA[15:8]	DATA[7:0]	
0	0	2 byte	A	A	*	*	
0	1	1 byte	A	N/A	*	---	Byte access to odd-numbered address
1	0	1 byte	N/A	A	---	*	Byte access to even-numbered address
1	1	1 byte	N/A	N/A	---	---	Note 2

Note 1. SHB*=0 and ADD[0]=1 are not output in the 8-bit device mode.

2. SHB*=1 and ADD[0]=1 are not output in the 16-bit device mode.

Remarks "A" indicates that the expansion bus outputs valid data.
"N/A" indicates that the expansion bus outputs invalid data.
"*" indicates the data that the expansion bus samples.
"---" indicates the data that the expansion bus does not samples.

(1) Operation of the expansion bus

The access time can be set from WISAA[2:0] (BCUSPEEDREG[6:4]).

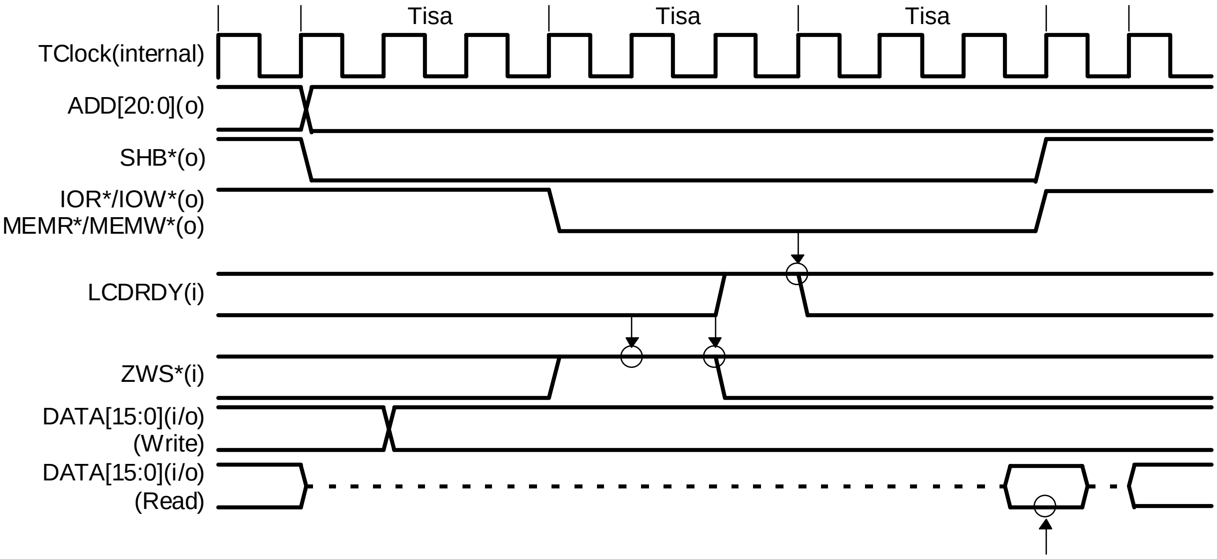
Table 10-26. Access Time of ISA

WISAA[2:0]	Tisa(TClock)
0 0 0	8
0 0 1	7
0 1 0	6
0 1 1	5
1 0 0	4
1 0 1	3
1 1 0	RFU
1 1 1	RFU

If the access time is set to 3 TClock (WISAA[2:0] = 101), the system bus brings bus cycle to an end at least 3 TClock (Tisa period) later the sampling of LCDRDY signal.

LCDRDY signal is sampled at the rising edge of TClock following the second and later Tisa period.

Figure 10-13. Two-Byte Access in the Case Where the LCDRDY High Level Is Sampled (WISAA[2:0] = 101)



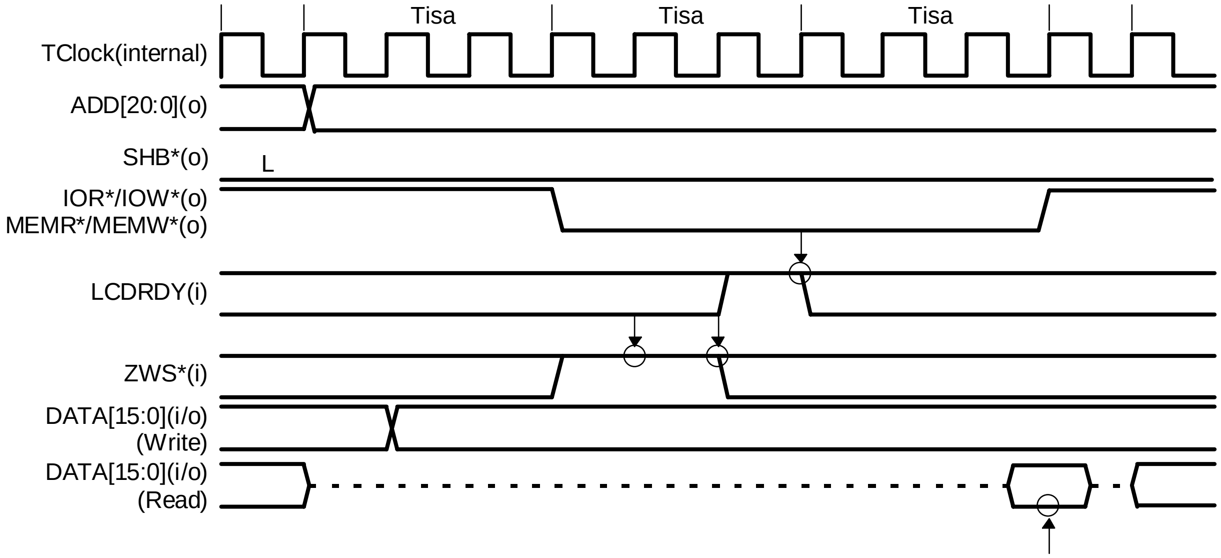
Remark Broken lines indicate high impedance.

Figure 10-14 indicates the timing at 1-byte access.

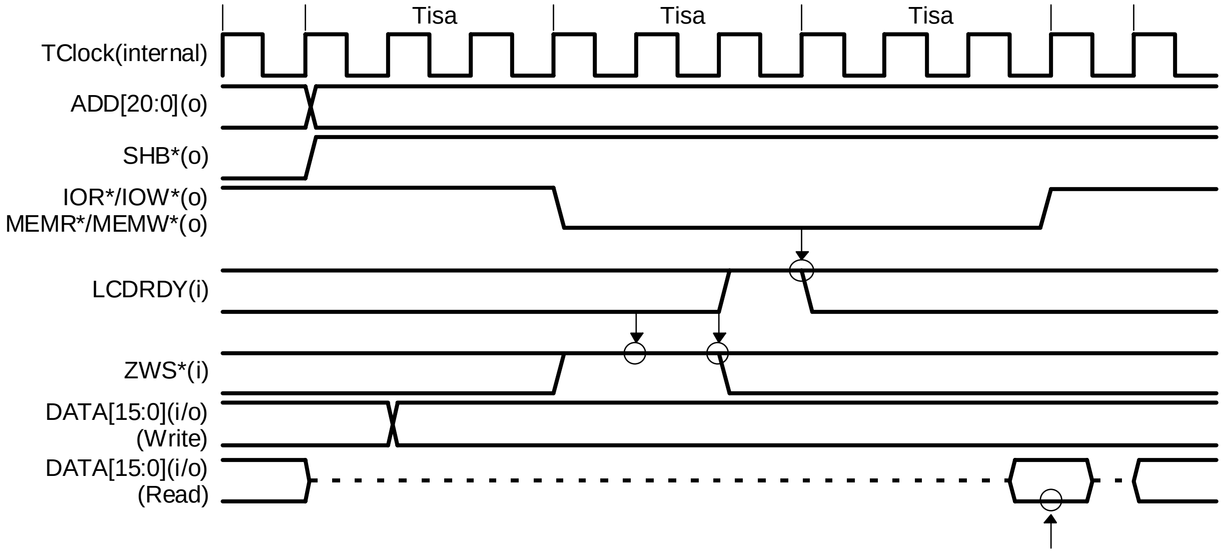
In the 16-bit device mode, low level is output from SHB* because the data access is performed using DATA[15:8]. In the 8-bit device mode, high level is output from SHB* because the data access is performed using DATA[7:0].

Figure 10-14. One-byte Access to Odd-numbered Address in the Case Where the LCDRDY High Level Is Sampled

(a) 16-bit device mode (WISAA[2:0] = 101)



(b) 8-bit device mode (WISAA[2:0] = 101)



Remark Broken lines indicate high impedance.

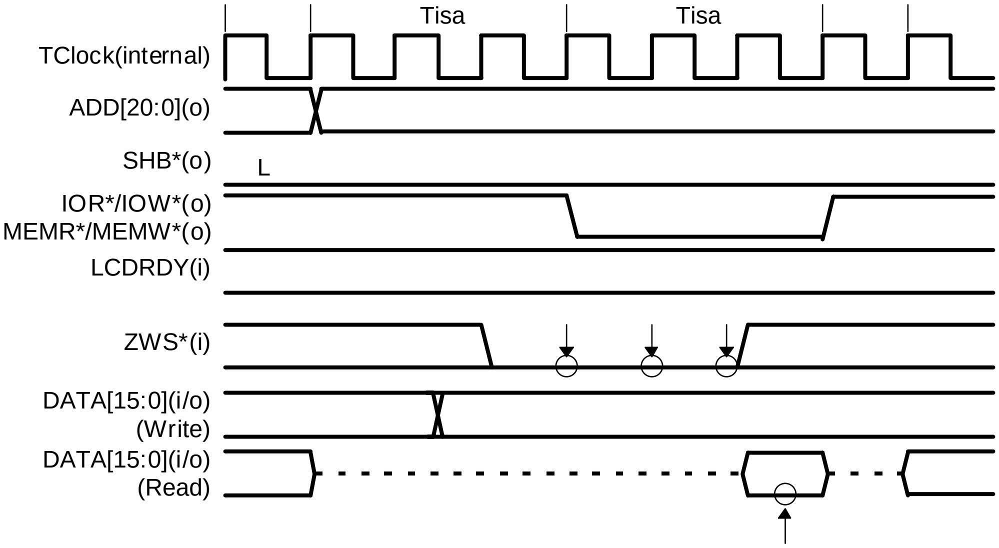
Figure 10-15 indicates the timing at 2-byte access when low level is sampled on ZWS*.

The bus cycle is brought to an end at least 1 TClock (Tisa period) later the sampling of ZWS* signal.

ZWS* is sampled at all of the rising edge of TClock following the second and later Tisa period.

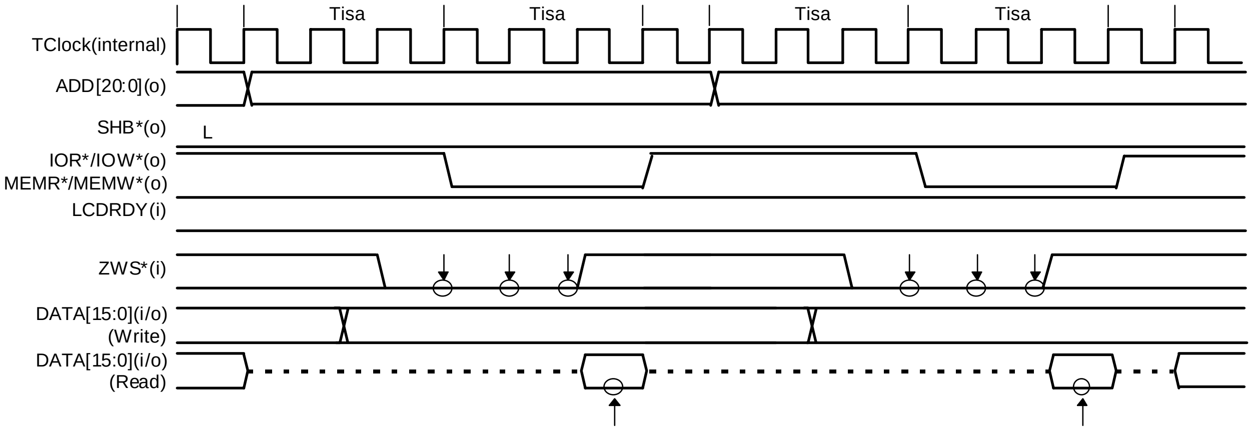
The system bus brings bus cycle to an end at least 1 TClock, or at most Tisa period (TClock number set in WISAA[2..0](BCUSPEEDREG[6..4])) later the sampling of LCDRDY signal.

Figure 10-15. Two-Byte Access in the Case Where the ZWS* Low Level Is Sampled (WISAA[2:0] = 101)



Remark Broken lines indicate high impedance.

Figure 10-16. Four-Byte Access in the Case Where the ZWS* Low Level Is Sampled (WISAA[2:0] = 101)



Remark Broken lines indicate high impedance.

10.6.3 LCD Interface

The access time can be set from WLCD[1:0] (BCUSPEEDREG[9:8]).

Table 10-27. Access Time of the LCD Interface

WLCD[1:0]	Tlcd(TClock)
0 0	8
0 1	6
1 0	4
1 1	2

Figure 10-17. Two-Byte Access to the LCD Controller (WLCDA[1:0] = 10)

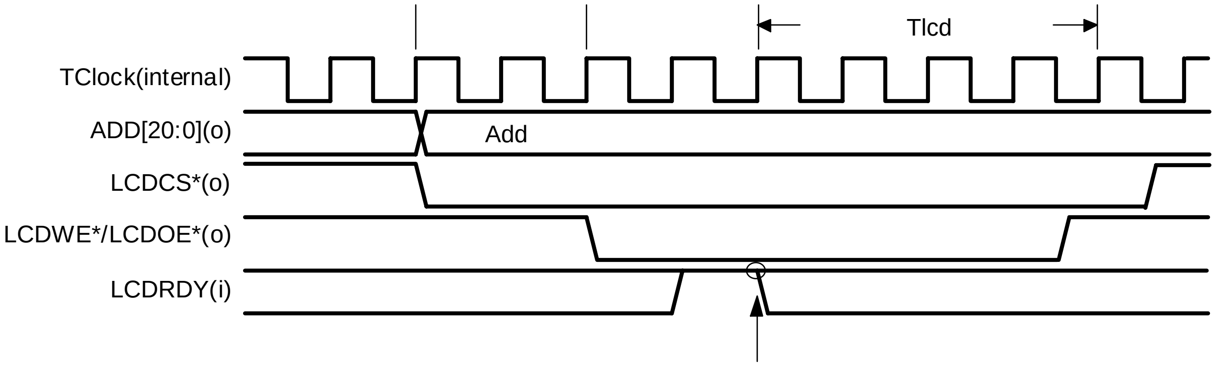
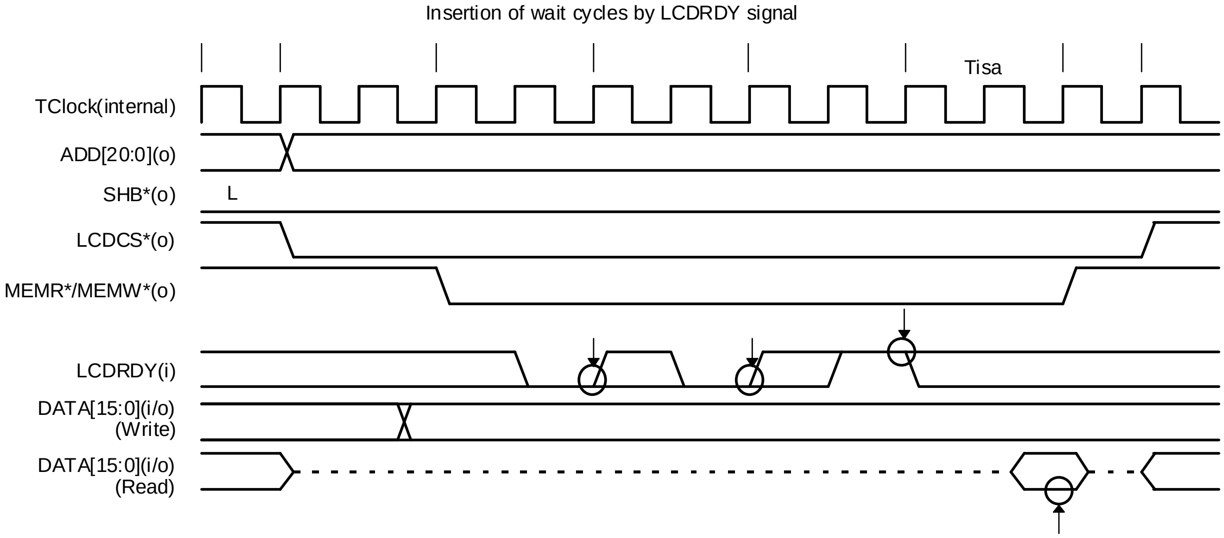


Figure 10-18. Two-Byte Access to the LCD Controller (WLCD[1:0] = 11)

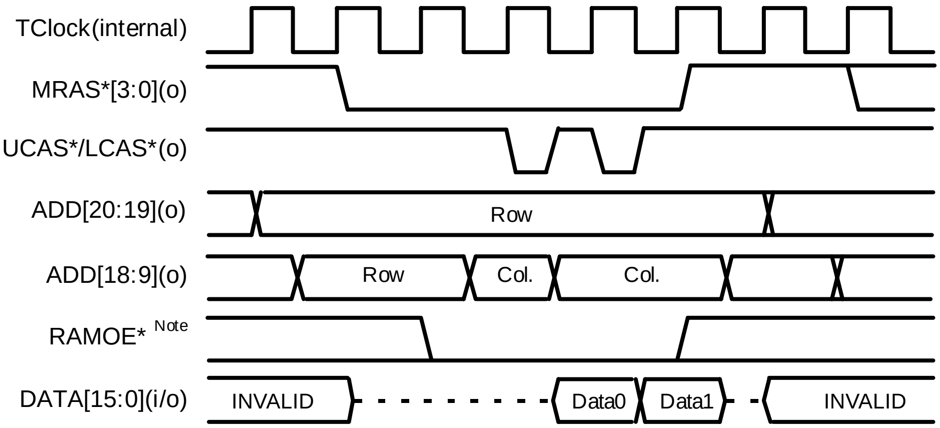


Remark Broken lines indicate high impedance.

10.6.4 DRAM Access (EDO type)

The access time to the DRAM is constant. Figures 10-19 and 10-20 indicate the timings of four-byte access to the DRAM.

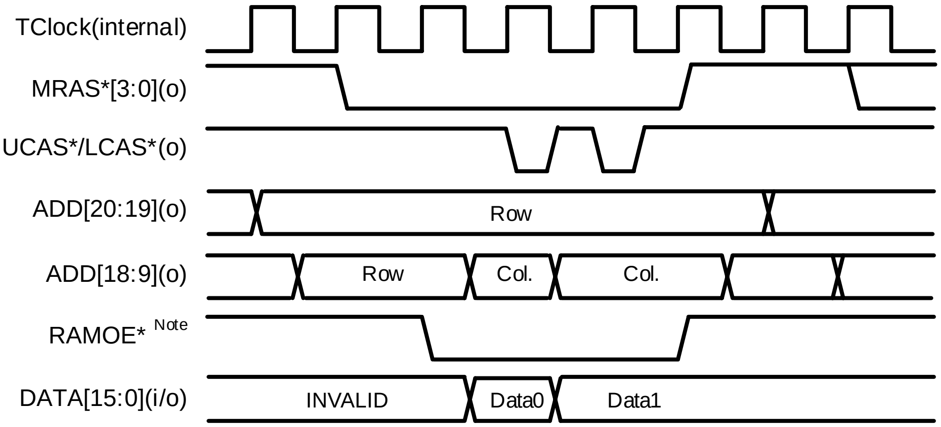
Figure 10-19. Four-Byte Read Access to the DRAM



Note The VR4101 has no output enable terminal for DRAM (RAMOE*). Generate RAMOE* signal by reversing the output of RSTSW* terminal.

Remark Broken lines indicate high impedance.

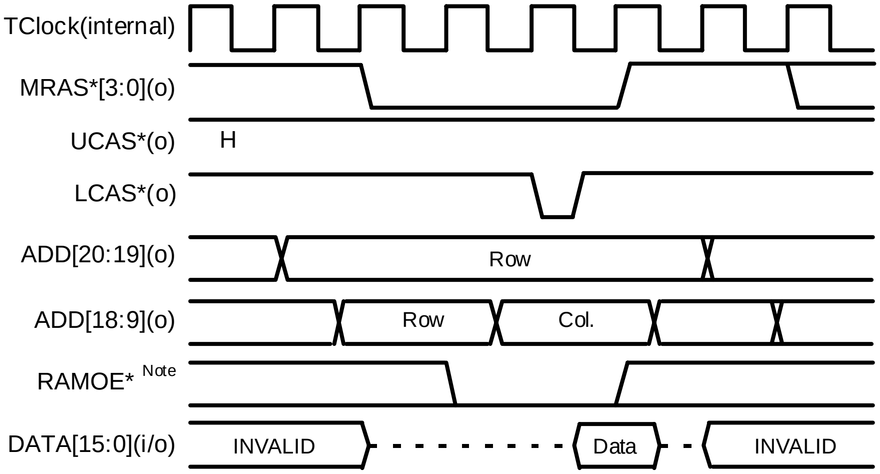
Figure 10-20. Four-Byte Write Access to the DRAM



Note The VR4101 has no output enable terminal for DRAM (RAMOE*). Generate RAMOE* signal by reversing the output of RSTSW* terminal.

Figures 10-21 to 10-24 indicate the timings of byte accesses to the DRAM.

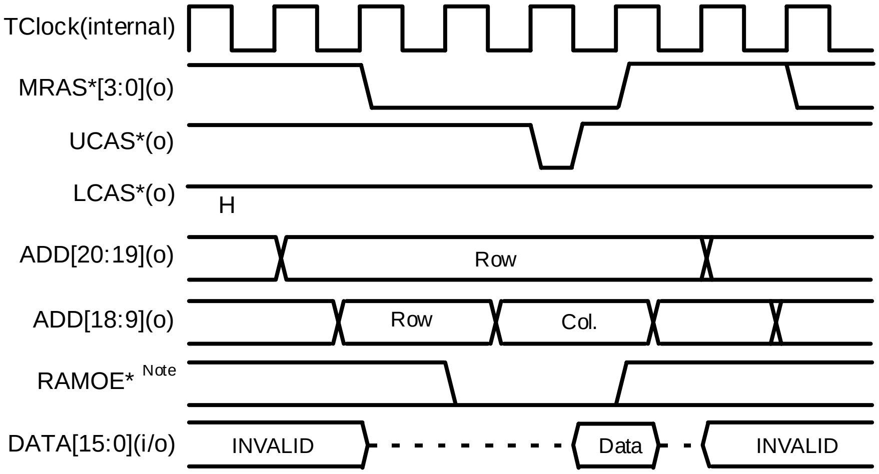
Figure 10-21. Byte Read from Odd-numbered Address of the DRAM



Note The VR4101 has no output enable terminal for DRAM (RAMOE*). Generate RAMOE* signal by reversing the output of RSTSW* terminal.

Remark Broken lines indicate high impedance.

Figure 10-22. Byte Read from Even-numbered Address of the DRAM



Note The VR4101 has no output enable terminal for DRAM (RAMOE*). Generate RAMOE* signal by reversing the output of RSTSW* terminal.

Remark Broken lines indicate high impedance.

Figure 10-23. Byte Write to Odd-numbered Address of the DRAM

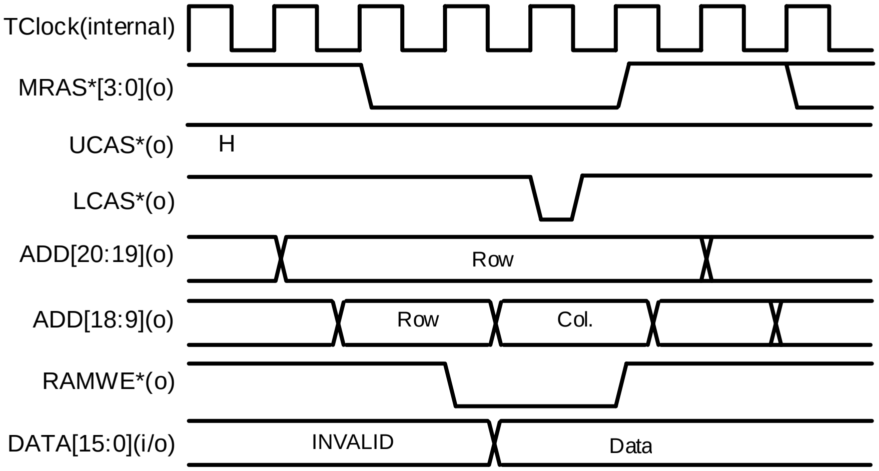
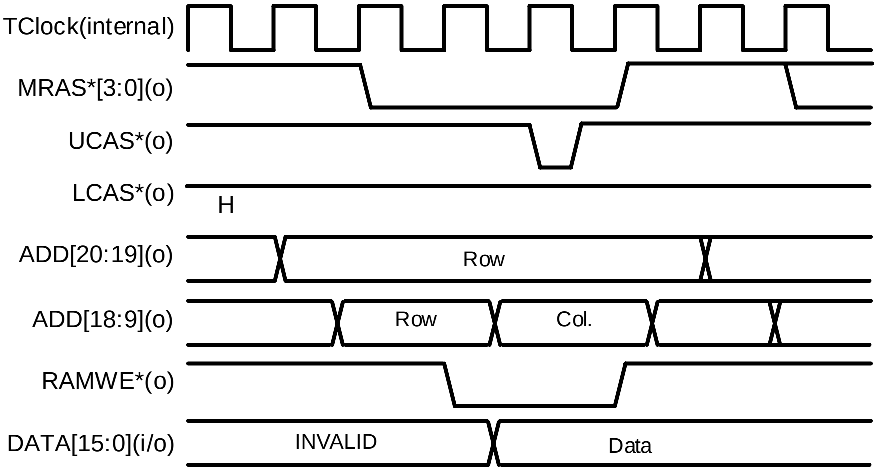


Figure 10-24. Byte Write to Even-numbered Address of the DRAM



10.6.5 Refresh

The VR4101 supports CBR refresh and self-refresh.

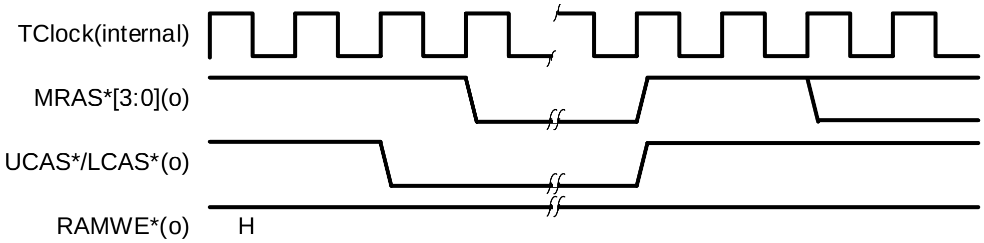
(1) CBR refresh

For CBR refresh, the refresh interval can be set by the REF1K bit of BCUCNTREG.

REF1K=1: CBR refresh is issued at intervals of 2059 TClock.

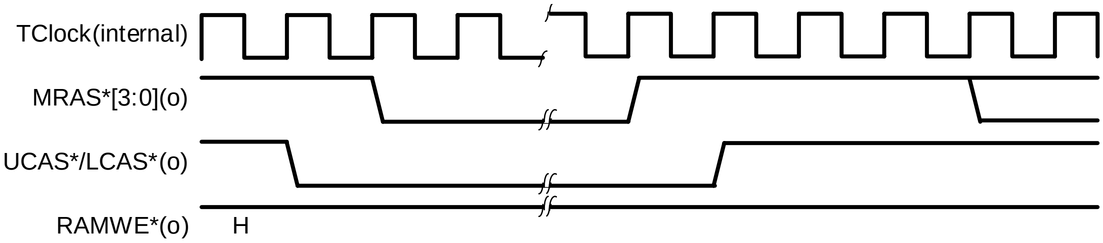
REF1K=0: CBR refresh is issued at intervals of 514 TClock.

Figure 10-25. CBR Refresh Cycle



(2) Self refresh

Figure 10-26. Self Refresh Cycle



CHAPTER 11 DMAAU (DMA ADDRESS UNIT)

This chapter explains the operation of the DMAAU and how to set the registers of the DMAAU.

11.1 GENERAL

The DMAAU controls the DMA addresses of PIU, SIU (transmission/reception), AIU, and KIU.

Any half-word address can be set as the DMA start address in a range of 0x0000 0000 to 0x001F FFFE. The DMA space is a 2-Kbyte space, aligned with a 2-Kbyte boundary, and which includes the DMA start address.

**Caution		If the DMA space for a peripheral unit is overlapped by that for another unit, the DMA operation is not guaranteed.**

11.2 REGISTER SET

The following table lists the registers of the DMAAU.

Table 11-1. DMAAU Registers

Address	R/W	Register symbols	Function
0x0B00 0020	R/W	PADDMAADRLREG	PAD1 DMA Address register Low
0x0B00 0022	R/W	PADDMAADRHREG	PAD1 DMA Address register High
0x0B00 0024	R/W	SRXDMAADRLREG	SRX1 DMA Address register Low
0x0B00 0026	R/W	SRXDMAADRHREG	SRX1 DMA Address register High
0x0B00 0028	R/W	STXDMAADRLREG	STX1 DMA Address register Low
0x0B00 002A	R/W	STXDMAADRHREG	STX1 DMA Address register High
0x0B00 002C	R/W	AUDDMAADRLREG	AUDIO1 DMA Address register Low
0x0B00 002E	R/W	AUDDMAADRHREG	AUDIO1 DMA Address register High
0x0B00 0030	R/W	KEYDMAADRLREG	KEY1 DMA Address register Low
0x0B00 0032	R/W	KEYDMAADRHREG	KEY1 DMA Address register High

The function of each of these registers is explained in detail below.

11.2.1 PADDMAADRLREG, PADDMAADRHREG

These registers set the base addresses of the DMA channel for the touch panel. Use a physical address value to set these registers.

Figure 11-1. PADDMAADRLREG (0x0B00 0020)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	PATDMA A[15]	PATDMA A[14]	PATDMA A[13]	PATDMA A[12]	PATDMA A[11]	PATDMA A[10]	PATDMA A[9]	PATDMA A[8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	PATDMA A[7]	PATDMA A[6]	PATDMA A[5]	PATDMA A[4]	PATDMA A[3]	PATDMA A[2]	PATDMA A[1]	PATDMA A[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..11]	PATDMA A[15..11]	DMA channel base address bits 15 through 11 for touch panel
D[10..1]	PATDMA A[10..1]	DMA channel offset address bits 10 through 1 for touch panel
D[0]	PATDMA A[0]	DMA channel offset address bit 0 for touch panel. Write 0 to this bit. 0 is returned when this bit is read.

Figure 11-2. PADDMAADRHREG (0x0B00 0022)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	PATDMA A[31]	PATDMA A[30]	PATDMA A[29]	PATDMA A[28]	PATDMA A[27]	PATDMA A[26]	PATDMA A[25]	PATDMA A[24]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	PATDMA A[23]	PATDMA A[22]	PATDMA A[21]	PATDMA A[20]	PATDMA A[19]	PATDMA A[18]	PATDMA A[17]	PATDMA A[16]
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	1	1	1	1

Bit position	Bit name	Function
D[15..5]	PATDMA A[31..21]	DMA channel base address bits 31 through 21 for touch panel. Write 0 to this bit. 0 is returned when this bit is read.
D[4..0]	PATDMA A[20..16]	DMA channel base address bits 20 through 16 for touch panel

11.2.2 SRXDMAADRLREG, SRXDMAADRHREG

These registers set the base addresses of the DMA channel for serial reception. Use a physical address value to set these registers.

Figure 11-3. SRXDMAADRLREG (0x0B00 0024)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	SRXDMA A[15]	SRXDMA A[14]	SRXDMA A[13]	SRXDMA A[12]	SRXDMA A[11]	SRXDMA A[10]	SRXDMA A[9]	SRXDMA A[8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SRXDMA A[7]	SRXDMA A[6]	SRXDMA A[5]	SRXDMA A[4]	SRXDMA A[3]	SRXDMA A[2]	SRXDMA A[1]	SRXDMA A[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..11]	SRXDMA A[15..11]	DMA channel base address bits 15 through 11 for serial reception
D[10..1]	SRXDMA A[10..1]	DMA channel offset address bits 10 through 1 for serial reception
D[0]	SRXDMA A[0]	DMA channel offset address bit 0 for serial reception. Write 0 to this bit. 0 is returned when this bit is read.

Figure 11-4. SRXDMAADRHREG (0x0B00 0026)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	SRXDMA A[31]	SRXDMA A[30]	SRXDMA A[29]	SRXDMA A[28]	SRXDMA A[27]	SRXDMA A[26]	SRXDMA A[25]	SRXDMA A[24]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SRXDMA A[23]	SRXDMA A[22]	SRXDMA A[21]	SRXDMA A[20]	SRXDMA A[19]	SRXDMA A[18]	SRXDMA A[17]	SRXDMA A[16]
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	1	1	1	1

Bit position	Bit name	Function
D[15..5]	SRXDMA A[31..21]	DMA channel base address bits 31 through 21 for serial reception. Write 0 to this bit. 0 is returned when this bit is read.
D[4..0]	SRXDMA A[20..16]	DMA channel base address bits 20 through 16 for serial reception

11.2.3 STXDMAADRLREG, STXDMAADRHREG

These registers set the base address of the DMA channel for serial transmission. Use a physical address value to set these registers.

Figure 11-5. STXDMAADRLREG (0x0B00 0028)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	STXDMA A[15]	STXDMA A[14]	STXDMA A[13]	STXDMA A[12]	STXDMA A[11]	STXDMA A[10]	STXDMA A[9]	STXDMA A[8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	0	1	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	STXDMA A[7]	STXDMA A[6]	STXDMA A[5]	STXDMA A[4]	STXDMA A[3]	STXDMA A[2]	STXDMA A[1]	STXDMA A[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..11]	STXDMA A[15..11]	DMA channel base address bits 15 through 11 for serial transmission
D[10..1]	STXDMA A[10..1]	DMA channel offset address bits 10 through 1 for serial transmission
D[0]	STXDMA A[0]	DMA channel offset address bit 0 for serial transmission. Write 0 to this bit. 0 is returned when this bit is read.

Figure 11-6. STXDMAADRHREG (0x0B00 002A)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	STXDM A A[31]	STXDM A A[30]	STXDM A A[29]	STXDM A A[28]	STXDM A A[27]	STXDM A A[26]	STXDM A A[25]	STXDM A A[24]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	STXDM A A[23]	STXDM A A[22]	STXDM A A[21]	STXDM A A[20]	STXDM A A[19]	STXDM A A[18]	STXDM A A[17]	STXDM A A[16]
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	1	1	1	1

Bit position	Bit name	Function
D[15..5]	STXDMA A[31..21]	DMA channel base address bits 31 through 21 for serial transmission. Write 0 to this bit. 0 is returned when this bit is read.
D[4..0]	STXDMA A[20..16]	DMA channel base address bits 20 through 16 for serial transmission

11.2.4 AUDDMAADRLREG, AUDDMAADRHREG

These registers set the base addresses of the DMA channel for audio output. Use a physical address value to set these registers.

Figure 11-7. AUDDMAADRLREG (0x0B00 002C)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	AUDDM A A[15]	AUDDM A A[14]	AUDDM A A[13]	AUDDM A A[12]	AUDDM A A[11]	AUDDM A A[10]	AUDDM A A[9]	AUDDM A A[8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	AUDDMA A[7]	AUDDMA A[6]	AUDDMA A[5]	AUDDMA A[4]	AUDDMA A[3]	AUDDMA A[2]	AUDDMA A[1]	AUDDMA A[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..11]	AUDDMA A[15..11]	DMA channel base address bits 15 through 11 for audio output
D[10..1]	AUDDMA A[10..1]	DMA channel offset address bits 10 through 1 for audio output
D[0]	AUDDMA A[0]	DMA channel offset address bit 0 for audio output. Write 0 to this bit. 0 is returned when this bit is read.

Figure 11-8. AUDDMAADRHREG (0x0B00 002E)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	AUDDMA A[31]	AUDDMA A[30]	AUDDMA A[29]	AUDDMA A[28]	AUDDMA A[27]	AUDDMA A[26]	AUDDMA A[25]	AUDDMA A[24]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	AUDDMA A[23]	AUDDMA A[22]	AUDDMA A[21]	AUDDMA A[20]	AUDDMA A[19]	AUDDMA A[18]	AUDDMA A[17]	AUDDMA A[16]
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	1	1	1	1

Bit position	Bit name	Function
D[15..5]	AUDDMA A[31..21]	DMA channel base address bits 31 through 21 for audio output. Write 0 to this bit. 0 is returned when this bit is read.
D[4..0]	AUDDMA A[20..16]	DMA channel base address bits 20 through 16 for audio output

11.2.5 KEYDMAADRLREG, KEYDMAADRHREG

These registers set the base addresses of the DMA channel for keyboard input. Use a physical address value to set these registers.

Figure 11-9. KEYDMAADRLREG (0x0B00 0030)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	KEYDM A A[15]	KEYDM A A[14]	KEYDM A A[13]	KEYDM A A[12]	KEYDM A A[11]	KEYDM A A[10]	KEYDM A A[9]	KEYDM A A[8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	0	1	1	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	KEYDM A A[7]	KEYDM A A[6]	KEYDM A A[5]	KEYDM A A[4]	KEYDM A A[3]	KEYDM A A[2]	KEYDM A A[1]	KEYDM A A[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..11]	KEYDMA A[15..11]	DMA channel base address bits 15 through 11 for keyboard input
D[10..1]	KEYDMA A[10..1]	DMA channel offset address bits 10 through 1 for keyboard input
D[0]	KEYDMA A[0]	DMA channel offset address bit 0 for keyboard input. Write 0 to this bit. 0 is returned when this bit is read.

Figure 11-10. KEYDMAADRHREG (0x0B00 0032)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	KEYDM A A[31]	KEYDM A A[30]	KEYDM A A[29]	KEYDM A A[28]	KEYDM A A[27]	KEYDM A A[26]	KEYDM A A[25]	KEYDM A A[24]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	KEYDM A A[23]	KEYDM A A[22]	KEYDM A A[21]	KEYDM A A[20]	KEYDM A A[19]	KEYDM A A[18]	KEYDM A A[17]	KEYDM A A[16]
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	1	1	1	1

Bit position	Bit name	Function
D[15..5]	KEYDMA A[31..21]	DMA channel base address bits 31 through 21 for keyboard input. Write 0 to this bit. 0 is returned when this bit is read.
D[4..0]	KEYDMA A[20..16]	DMA channel base address bits 20 through 16 for keyboard input

[MEMO]

CHAPTER 12 DCU (DMA CONTROL UNIT)

This chapter explains the operation of the DCU and how to set the registers of the DCU.

12.1 GENERAL

The DCU performs DMA control. It controls DMA requests received from each internal peripheral I/O unit (such as SIU, KIU, PIU, and AIU) and acknowledge signals received from the BCU that arbitrates the bus, and enables or disables DMA.

When DMA requests of built-in peripheral I/O units are received concurrently, the DCU processes such DMA requests according to the following priority order. This priority order cannot be changed.

Table 12-1. Priority Order of DMAs

Priority order	Type of DMA
High	Audio output
	Touch-panel input
	Serial receiving
	Serial transmission
Low	Keyboard input

12.2 REGISTER SET

The following table lists the registers of the DCU.

Table 12-2. DCU Registers

Address	R/W	Register symbols	Function
0x0B00 0040	R/W	DMARSTREG	DMA Reset register
0x0B00 0042	R/W	DMAIDLEREG	DMA Idle register
0x0B00 0044	R/W	DMASENREG	DMA Sequencer Enable register
0x0B00 0046	R/W	DMAMSKREG	DMA Mask register
0x0B00 0048	R/W	DMAREQREG	DMA Request register

The function of each of these registers is explained in detail below.

12.2.1 DMARSTREG

Figure 12-1. DMARSTREG (0x0B00 0040)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	DMARST
R/W	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	DMARST	Resets DMA controller. 1: Reset 0: Normal

This register is used to initialize the DMA controller.

12.2.2 DMAIDLEREG

Figure 12-2. DMAIDLEREG (0x0B00 0042)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	DMAI STAT
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	DMAISTAT	Indicates DMA sequencer status. 1: D_IDLE status 0: DMA used

This register indicates the status of the DMA sequencer.

12.2.3 DMASENREG

Figure 12-3. DMASENREG (0x0B00 0044)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	DMASEN
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	DMASEN	Enables DMA sequencer. 1: Enabled 0: Disabled

This register enables or disables the DMA sequencer.

12.2.4 DMAMSKREG

Figure 12-4. DMAMSKREG (0x0B00 0046)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	DMAMSK KIU	DMAMSK ADU	DMAMSK STX	DMAMSK SRX	DMAMSK PIU
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..5]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[4]	DMAMSKKIU	Enables KIU DMA transfer. 1: Enabled 0: Disabled
D[3]	DMAMSKADU	Enables AIU DMA transfer. 1: Enabled 0: Disabled
D[2]	DMAMSKSTX	Enables SIU transmission DMA transfer. 1: Enabled 0: Disabled
D[1]	DMAMSKSRX	Enables SIU reception DMA transfer. 1: Enabled 0: Disabled
D[0]	DMAMSKPIU	Enables PIU DMA transfer. 1: Enabled 0: Disabled

This register enables or disables each DMA transfer.

Set each DMA transfer enable bit when the DMA sequencer is in the D_IDLE status. Otherwise, the operation of the VR4101 will be undefined.

12.2.5 DMAREQREG

Figure 12-5. DMAREQREG (0x0B00 0048)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	DRQKIU	DRQADU	DRQSTX	DRQSRX	DRQPIU
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..5]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[4]	DRQKIU	KIU DMA transfer request 1: Requested 0: Not requested
D[3]	DRQADU	AIU DMA transfer request 1: Requested 0: Not requested
D[2]	DRQSTX	SIU transmission DMA transfer request 1: Requested 0: Not requested
D[1]	DRQSRX	SIU reception DMA transfer request 1: Requested 0: Not requested
D[0]	DRQPIU	PIU DMA transfer request 1: Requested 0: Not requested

This register indicates the presence or absence of DMA transfer request.

CHAPTER 13 CMU (CLOCK MASK UNIT)

This chapter explains the operation of the CMU and how to set the registers of the CMU.

13.1 GENERAL

This unit enables to reduce the power consumption of unused units by providing a masking measure when the input clock from of the CPU (I_tclk) is supplied to each unit. Object units include KIU, PIU, GIU, SIU, AIU, DebugSIU and RTC.

The functions of the internal blocks in the CMU are summarized as follows:

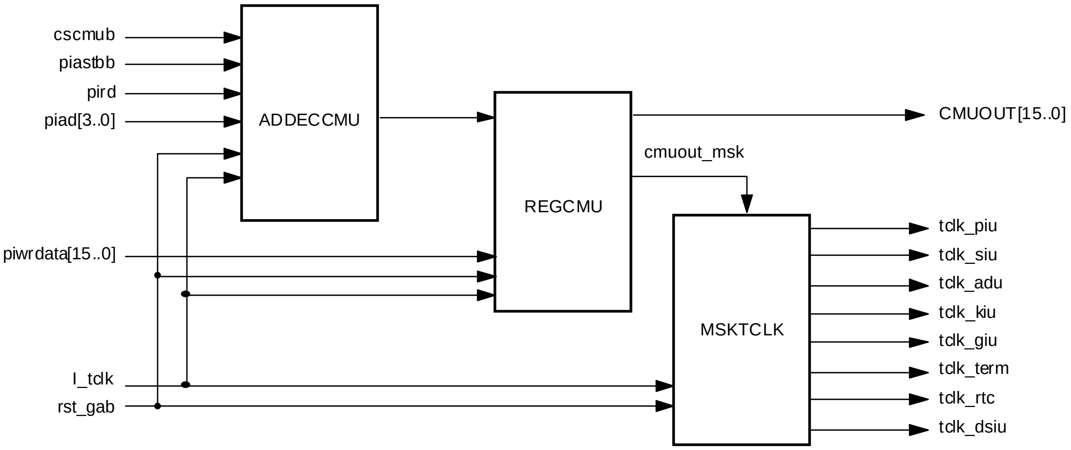
ADDECCMU ....The address decoder for read/write access from the CPU to the register.

REGCMU ....Has the register for clock masking.
initial=0=mask. Clock is not supplied unless the CPU performs write (1) to the register.

MSKTCLK ....Mask unit for TClock. Has FF and AND that operate in synchronization with the falling edge of TClock.

A block diagram of the CMU is shown below.

Figure 13-1. Block Diagram of the CMU



13.2 REGISTER SET

The following table lists the registers of the CMU.

Table 13-1. CMU Register

Address	R/W	Register symbols	Function
0x0B00 0060	R/W	CMUCLKMSKREG	CMU Clock Mask register

The function of this register is explained in detail below.

13.2.1 CMUCLKMSKREG

Figure 13-2. CMUCLKMSKREG (0x0B00 0060)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	MSKRTC	MSKDSIU	MSKGIU	MSKKIU	MSKADU	MSKSIU	MSKPIU
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..7]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[6]	MSKRTC	RTC clock supply 1: Supplied 0: Masked
D[5]	MSKDSIU	Debug SIU clock supply 1: Supplied 0: Masked
D[4]	MSKGIU	GIU clock supply 1: Supplied 0: Masked
D[3]	MSKKIU	KIU clock supply 1: Supplied 0: Masked
D[2]	MSKADU	AIU clock supply 1: Supplied 0: Masked
D[1]	MSKSIU	SIU clock supply 1: Supplied 0: Masked
D[0]	MSKPIU	PIU clock supply 1: Supplied 0: Masked

This register masks the supply of the clock to RTC, DebugSIU, GIU, KIU, AIU, SIU, and PIU.

[MEMO]

CHAPTER 14 ICU (INTERRUPT CONTROL UNIT)

This chapter explains the operation of the ICU and how to set the registers of the ICU.

14.1 GENERAL

The ICU summarizes interrupt signals from each built-in peripheral unit and transfers interrupt signals (Int0, Int1, NMI) to the CPU CORE.

The functions of the ICU are outlined below.

ADDECICU.............Performs address decode of the read/write access from the CPU to the registers in the ICU.

REGICU.................Has the register for clock masking.
initial=0=mask. Clock is not supplied unless the CPU performs write (1) to the register.

OUTICU ....Performs summarization after masking each interrupt (all outputs are synchronized with the rising edge of I_mclkin). Further, controls the masking of interrupts during the setting in the Suspend mode (doze_mskint), assert of the int_all signal, interrupting factor summarizing signal, and the memdrv assert timing signal at the restoration from the Suspend mode.

Interrupt requests to the CPU core are noticed by using following three signals:

NMI: battint_intr alone.

However, switching between NMI and Int0 can be enabled by the setting on the register. Switch to Int0 if a user intends to mask battint_intr, since NMI cannot be controlled with the masking of interrupt by means of software.

Int1:   rtc_long_intr alone.

This is exclusively used because interrupt factors such as Interval Timer require a quicker response than that of other interrupt factors.

Int0:   All other interrupts.

Refer to 14.2 for the details of interrupt factors.

How an interrupt request is notified to the CPU core is shown below.

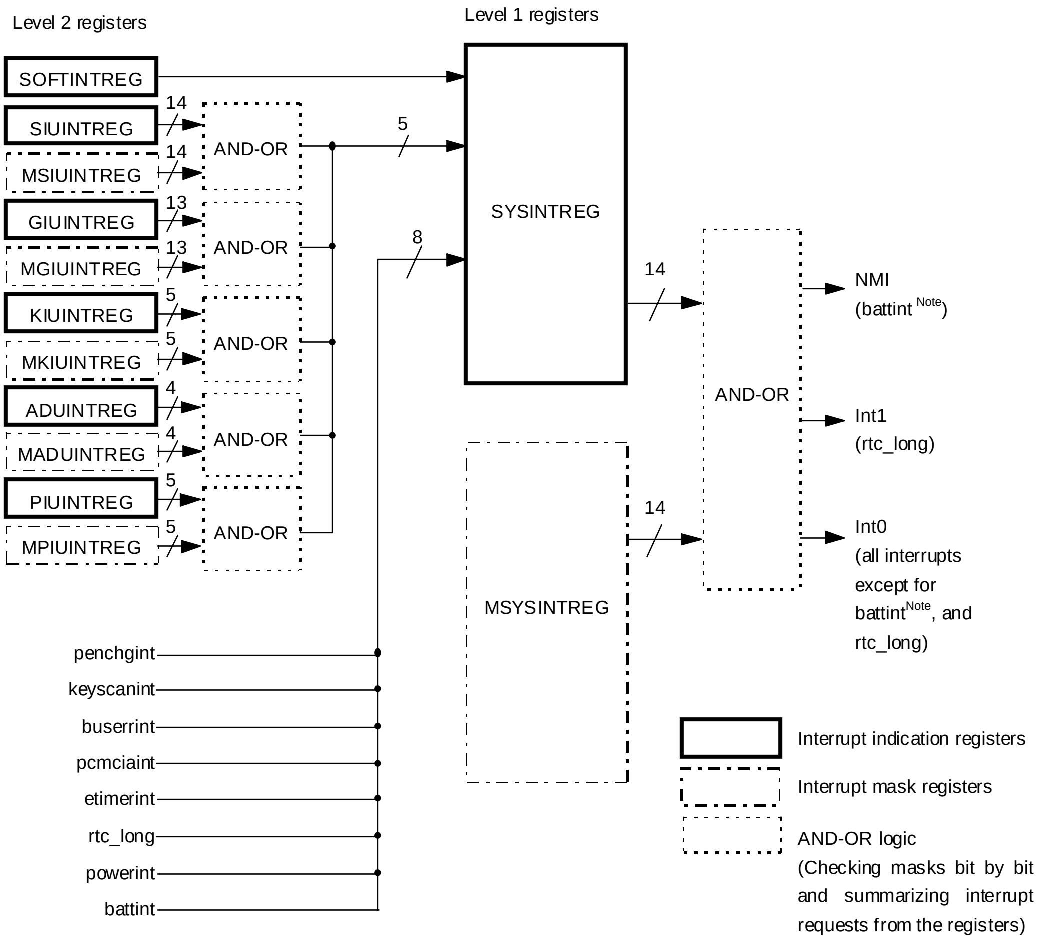
If an interrupt request occurs in the peripheral units, the corresponding bit in the interrupt indication register of Level 2 (xxxINTREG) is set to 1. The interrupt indication register is ANDed bit-wise with the corresponding interrupt mask register of Level 2 (MxxxINTREG). If the occurred interrupt request is enabled (set to 1) in the mask register, the interrupt request is notified to the interrupt indication register of Level 1 (SYSINTREG) and the corresponding bit is set to 1. At this time, the interrupt requests from the same register of Level 2 are notified to the SYSINTREG as a single interrupt request.

Interrupt requests from some units directly set their corresponding bits in the SYSINTREG.

The SYSINTREG is ANDed bit-wise with the interrupt mask register of Level 1 (MSYSINTREG). If the interrupt request is enabled (set to 1) in MSYSINTREG, a corresponding interrupt request signal is output from the ICU to the CPU core. battint is connected to the NMI or Int0 signal of the CPU core (selected by setting of NMIREG). rtc_long is connected to the Int1 signal of the CPU core. The other interrupt requests are connected to the Int0 signal of the CPU core as a one interrupt request.

The following figure shows an outline of interrupt control in the ICU.

Figure 14-1. Outline of Interrupt Control



Note Which of NMI and Int0 is used for battint is selectable by setting of NMIREG.

14.2 REGISTER SET

The following table lists the registers of the ICU.

Table 14-1. ICU Registers

Address	R/W	Register symbols	Function
0x0B00 0080	R	SYSINTREG	Level 1 System register
0x0B00 0082	R	PIUINTRTG	Level 2 PIU register
0x0B00 0084	R	ADUINTREG	Level 2 AIU register
0x0B00 0086	R	KIUINTREG	Level 2 KIU register
0x0B00 0088	R	GIUINTREG	Level 2 GIU register
0x0B00 008A	R	SIUINTREG	Level 2 SIU register
0x0B00 008C	R/W	MSYSINTREG	Level 1 Mask System register
0x0B00 008E	R/W	MPIUINTRTG	Level 2 Mask PIU register
0x0B00 0090	R/W	MADUINTREG	Level 2 Mask AIU register
0x0B00 0092	R/W	MKIUINTREG	Level 2 Mask KIU register
0x0B00 0094	R/W	MGIUINTREG	Level 2 Mask GIU register
0x0B00 0096	R/W	MSIUINTREG	Level 2 Mask SIU register
0x0B00 0098	R/W	NMIREG	NMI selection register
0x0B00 009A	R/W	SOFTINTREG	Software Interrupt register

The function of each of these registers is explained in detail below.

14.2.1 Level-1 System Register

Figure 14-2. SYSINTREG (0x0B00 0080) (1/2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	DOZE PIUINTR	DOZE KIUINTR	SOFT INTR	WRBERR INTR	SIUINTR	GIUINTR
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	KIUINTR	ADUINTR	PIUINTR	PCMCIA INTR	ETIMER INTR	RTCL INTR	POWER INTR	BATINTR
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	DOZEPIUINTR	PIU interrupt in Suspend mode 1: Occurred 0: Normal
D[12]	DOZEKIUINTR	KIU interrupt in Suspend mode 1: Occurred 0: Normal
D[11]	SOFTINTR	Software interrupt (generated by setting SOFTINTREG) 1: Occurred 0: Normal
D[10]	WRBERRINTR	Bus error interrupt 1: Occurred 0: Normal
D[9]	SIUINTR	SIU interrupt 1: Occurred 0: Normal
D[8]	GIUINTR	GIU interrupt 1: Occurred 0: Normal
D[7]	KIUINTR	KIU interrupt 1: Occurred 0: Normal

Figure 14-2. SYSINTREG (0x0B00 0080) (2/2)

Bit position	Bit name	Function
D[6]	ADUINTR	AIU interrupt 1: Occurred 0: Normal
D[5]	PIUINTR	PIU interrupt 1: Occurred 0: Normal
D[4]	PCMCIAINTR	PCMCIA interrupt 1: Occurred 0: Normal
D[3]	ETIMERINTR	ETIMER interrupt 1: Occurred 0: Normal
D[2]	RTCLINTR	RTCLong interrupt 1: Occurred 0: Normal
D[1]	POWERINTR	Power SW interrupt 1: Occurred 0: Normal
D[0]	BATINTR	Battery interrupt 1: Occurred 0: Normal

This register indicates the occurrence of each interrupt of the VR4101 system.

14.2.2 Level-2 PIU Register

Figure 14-3. PIUINTREG (0x0B00 0082)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	PADEND INTR	PADINTR	PADDLO STINTR	PADDRD YINTR	PADCHG INTR
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..5]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[4]	PADENDINTR	PIU DMA transfer 2-page boundary interrupt 1: Occurred 0: Normal
D[3]	PADINTR	PIU DMA transfer 1-page boundary interrupt 1: Occurred 0: Normal
D[2]	PADDLOSTINTR	PIU data lost interrupt 1: Occurred 0: Normal
D[1]	PADDRDYINTR	PIU DMA transfer end interrupt 1: Occurred 0: Normal
D[0]	PADCHGINTR	Touch panel contact status change interrupt 1: Occurred 0: Normal

This register indicates the occurrence of each interrupt of the PIU.

14.2.3 Level-2 AIU Register

Figure 14-4. ADUINTREG (0x0B00 0084)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	AIUEND INTR	AIUINTR	AIUIDLE INTR	AIUST INTR
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..4]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[3]	AIUENDINTR	AIU DMA transfer 2-page boundary interrupt 1: Occurred 0: Normal
D[2]	AIUINTR	AIU DMA transfer 1-page boundary interrupt 1: Occurred 0: Normal
D[1]	AIUIDLEINTR	AIU sequencer Idle interrupt 1: Occurred 0: Normal
D[0]	AIUSTINTR	AIU sequencer operation start interrupt 1: Occurred 0: Normal

This register indicates the occurrence of each interrupt of the AIU.

14.2.4 Level-2 KIU Register

Figure 14-5. KIUINTREG (0x0B00 0086)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	KEYEND INTR	KEYINTR	KEYDATL OSTINTR	KEYDAT RDYINTR	KEYSCAN INTR
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..5]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[4]	KEYENDINTR	KIU DMA transfer 2-page boundary interrupt 1: Occurred 0: Normal
D[3]	KEYINTR	KIU DMA transfer 1-page boundary interrupt 1: Occurred 0: Normal
D[2]	KEYDATLOST INTR	Key data scan lost interrupt 1: Occurred 0: Normal
D[1]	KEYDATRDY INTR	Key data scan end interrupt 1: Occurred 0: Normal
D[0]	KEYSCANINT R	Key input detection interrupt 1: Occurred 0: Normal

This register indicates the occurrence of each interrupt of the KIU.

14.2.5 Level-2 GIU Register

Figure 14-6. GIUINTREG (0x0B00 0088)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	IOINTR [13]	Reserved	IOINTR [11]	IOINTR [10]	IOINTR [9]	IOINTR [8]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IOINTR [7]	IOINTR [6]	IOINTR [5]	IOINTR [4]	IOINTR [3]	IOINTR [2]	IOINTR [1]	IOINTR [0]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	IOINTR[13]	DCD pin interrupt 1: Occurred 0: Normal
D[12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IOINTR[11..0]	GPIO[11..0] pins interrupt 1: Occurred 0: Normal

This register indicates the occurrence of each interrupt of the GIU.

14.2.6 Level-2 SIU Register

Figure 14-7. SIUINTREG (0x0B00 008A) (1/2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	INTSER0	INTSR0	INTST0	BR	FE	DCD
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0

Name	DSR	CTS	RXL	RXG	RXE	RXI	TXE	TXI
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	INTSER0	Debug serial reception error interrupt 1: Occurred 0: Normal
D[12]	INTSR0	Debug serial reception end interrupt 1: Occurred 0: Normal
D[11]	INTST0	Debug serial transmission end interrupt 1: Occurred 0: Normal
D[10]	BR	Break signal detection interrupt 1: Occurred 0: Normal
D[9]	FE	Framing error detection interrupt 1: Occurred 0: Normal
D[8]	DCD	DCD signal detection interrupt 1: Occurred 0: Normal
D[7]	DSR	DSR* signal detection interrupt 1: Occurred 0: Normal
D[6]	CTS	CTS* signal detection interrupt 1: Occurred 0: Normal

Figure 14-7. SIUINTREG (0x0B00 008A) (2/2)

Bit position	Bit name	Function
D[5]	RXL	1-character reception lost detection interrupt 1: Occurred 0: Normal
D[4]	RXG	1-character reception end detection interrupt 1: Occurred 0: Normal
D[3]	RXE	Reception data DMA transfer 2-page boundary interrupt 1: Occurred 0: Normal
D[2]	RXI	Reception data DMA transfer 1-page boundary interrupt 1: Occurred 0: Normal
D[1]	TXE	Transmission data DMA transfer 2-page boundary interrupt 1: Occurred 0: Normal
D[0]	TXI	Transmission data DMA transfer 1-page boundary interrupt 1: Occurred 0: Normal

This register indicates the occurrence of each interrupt of SIU and Debug SIU.

14.2.7 Level-1 Mask System Register

Figure 14-8. MSYSINTREG (0x0B00 008C) (1/2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	DOZE PIUINTR	DOZE KIUINTR	SOFT INTR	WRBERR INTR	SIUINTR	GIUINTR
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	KIUINTR	ADUINTR	PIUINTR	PCMCIA INTR	ETIMER INTR	RTCL INTR	POWER INTR	BATINTR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	DOZEPIUINTR	Enables PIU interrupt in Suspend mode. 1: Enabled 0: Disabled
D[12]	DOZEKIUINTR	Enables KIU interrupt in Suspend mode. 1: Enabled 0: Disabled
D[11]	SOFTINTR	Enables software interrupt. 1: Enabled 0: Disabled
D[10]	WRBERRINTR	Enables bus error interrupt. 1: Enabled 0: Disabled
D[9]	SIUINTR	Enables SIU interrupt. 1: Enabled 0: Disabled
D[8]	GIUINTR	Enables GIU interrupt. 1: Enabled 0: Disabled
D[7]	KIUINTR	Enables KIU interrupt. 1: Enabled 0: Disabled
D[6]	ADUINTR	Enables AIU interrupt. 1: Enabled 0: Disabled

Figure 14-8. MSYSINTREG (0x0B00 008C) (2/2)

Bit position	Bit name	Function
D[5]	PIUINTR	Enables PIU interrupt. 1: Enabled 0: Disabled
D[4]	PCMCIAINTR	Enables PCMCIA interrupt 1: Enabled 0: Disabled
D[3]	ETIMERINTR	Enables ETIMER interrupt 1: Enabled 0: Disabled
D[2]	RTCLINTR	Enables RTCLong interrupt 1: Enabled 0: Disabled
D[1]	POWERINTR	Enables Power SW interrupt 1: Enabled 0: Disabled
D[0]	BATINTR	Enables Battery interrupt 1: Enabled 0: Disabled

This register is used to mask each interrupt of the VR4101 system.

14.2.8 Level-2 Mask PIU Register

Figure 14-9. MPIUINTREG (0x0B00 008E)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	PADEND INTR	PADINTR	PADDLO STINTR	PADDRD YINTR	PADCHG INTR
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..5]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[4]	PADENDINTR	Enables PIU DMA transfer 2-page boundary interrupt. 1: Enabled 0: Disabled
D[3]	PADINTR	Enables PIU DMA transfer 1-page boundary interrupt. 1: Enabled 0: Disabled
D[2]	PADDLOSTINTR	Enables PIUDATAREG data overwrite interrupt. 1: Enabled 0: Disabled
D[1]	PADDRDYINTR	Enables PIU DMA transfer end interrupt. 1: Enabled 0: Disabled
D[0]	PADCHGINTR	Enables touch panel contact status change interrupt. 1: Enabled 0: Disabled

This register is used to mask each interrupt of the PIU.

14.2.9 Level-2 Mask AIU Register

Figure 14-10. MADUINTREG (0x0B00 0090)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	AIUEND INTR	AIUINTR	AIUIDLE INTR	AIUST INTR
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..4]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[3]	AIUENDINTR	Enables AIU DMA transfer 2-page boundary interrupt. 1: Enabled 0: Disabled
D[2]	AIUINTR	Enables AIU DMA transfer 1-page boundary interrupt. 1: Enabled 0: Disabled
D[1]	AIUIDLEINTR	Enables AIU sequencer Idle interrupt. 1: Enabled 0: Disabled
D[0]	AIUSTINTR	Enables AIU sequencer operation start interrupt. 1: Enabled 0: Disabled

This register is used to mask each interrupt of the AIU.

14.2.10 Level-2 Mask KIU Register

Figure 14-11. MKIUINTREG (0x0B00 0092)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	KEYEND INTR	KEYINTR	KEYDATL OSTINTR	KEYDAT RDYINTR	KEYSCA NINTR
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..5]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[4]	KEYENDINTR	Enables KIU DMA transfer 2-page boundary interrupt. 1: Enabled 0: Disabled
D[3]	KEYINTR	Enables KIU DMA transfer 1-page boundary interrupt. 1: Enabled 0: Disabled
D[2]	KEYDATLOST INTR	Enables key scan data lost interrupt. 1: Enabled 0: Disabled
D[1]	KEYDATRDY INTR	Enables key data scan end interrupt. 1: Enabled 0: Disabled
D[0]	KEYSCANINTR	Enables key input detection interrupt. 1: Enabled 0: Disabled

This register is used to mask each interrupt of the KIU.

14.2.11 Level-2 Mask GIU Register

Figure 14-12. MGIUINTREG (0x0B00 0094)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	IOINTR [13]	Reserved	IOINTR [11]	IOINTR [10]	IOINTR [9]	IOINTR [8]
R/W	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IOINTR [7]	IOINTR [6]	IOINTR [5]	IOINTR [4]	IOINTR [3]	IOINTR [2]	IOINTR [1]	IOINTR [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	IOINTR[13]	Enables DCD pin interrupt. 1: Enabled 0: Disabled
D[12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IOINTR[11..0]	Enables GPIO[11..0] pins interrupt. 1: Enabled 0: Disabled

This register is used to mask each interrupt of the GIU.

14.2.12 Level-2 Mask SIU Register

Figure 14-13. MSIUINTREG (0x0B00 0096) (1/2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	INTSER0	INTSR0	INTST0	BR	FE	DCD
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	DSR	CTS	RXL	RXG	RXE	RXI	TXE	TXI
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	INTSER0	Enables debug serial reception error interrupt. 1: Enabled 0: Disabled
D[12]	INTSR0	Enables debug serial reception end interrupt. 1: Enabled 0: Disabled
D[11]	INTST0	Enables debug serial transfer end interrupt. 1: Enabled 0: Disabled
D[10]	BR	Enables break signal detection interrupt. 1: Enabled 0: Disabled
D[9]	FE	Enables framing error detection interrupt. 1: Enabled 0: Disabled
D[8]	DCD	Enables DCD signal detection interrupt. 1: Enabled 0: Disabled
D[7]	DSR	Enables DSR* signal detection interrupt. 1: Enabled 0: Disabled
D[6]	CTS	Enables CTS* signal detection interrupt. 1: Enabled 0: Disabled

Figure 14-13. MSIUINTREG (0x0B00 0096) (2/2)

Bit position	Bit name	Function
D[5]	RXL	Enables 1-character reception lost detection interrupt. 1: Enabled 0: Disabled
D[4]	RXG	Enables 1-character reception end detection interrupt. 1: Enabled 0: Disabled
D[3]	RXE	Enables receive data DMA transfer 2-page boundary interrupt. 1: Enabled 0: Disabled
D[2]	RXI	Enables receive data DMA transfer 1-page boundary interrupt. 1: Enabled 0: Disabled
D[1]	TXE	Enables transfer data DMA transfer 2-page boundary interrupt. 1: Enabled 0: Disabled
D[0]	TXI	Enables transfer data DMA transfer 1-page boundary interrupt. 1: Enabled 0: Disabled

This register is used to mask each interrupt of the SIU and Debug SIU.

14.2.13 NMI Register

Figure 14-14. NMIREG (0x0B00 0098)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	NMIOR INT
R/W	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	NMIORINT	Sets type of low battery voltage detection interrupt. 1: Int0 0: NMI

This register is used to set the type of the interrupt reported to the VR4100 CPU core if a low battery voltage detection interrupt occurs.

14.2.14 Software Interrupt Register

Figure 14-15. SOFTINTREG (0x0B00 009A)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	SOFT INTR
R/W	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	SOFTINTR	Sets software interrupt. 1: Set 0: Clear

This register is used to generate the software interrupt.

14.3 NOTES FOR REGISTER SETTING

For the ICU, there is no special register setting flow.

The interrupt mask register is set to initial=0=Mask immediately when it has been started up. Therefore, the CPU must release the masks on those interrupts that are required for the startup process without fail (Release of the mask on battint_intr = NMI is required to be effected at all times).

battint_intr is set to initial = 0 = NMI. To switch it to Int0, it is required to write 1 to the NMIREG.

soft_intr is a software interrupt. Interrupt Int0 is asserted only with the write to the SOFTINTREG. Interrupt is cleared by writing 0.

[MEMO]

CHAPTER 15 PMU (POWER MANAGEMENT UNIT)

This chapter explains the operation of the PMU and how to set the registers of the PMU.

15.1 GENERAL

The PMU controls the internal power consumption of the VR4101 as well as the power consumption of a system configured around the VR4101.

- ✧ Reset control
- ✧ Shutdown control
- ✧ Power-ON sequence control
- ✧ Low power consumption mode control

15.1.1 Reset Control

A list of the RTC, peripheral unit, CPU core, and the bit of PMUINTREG to be set during a reset is shown below.

Table 15-1. Types of Reset and Processor Status

Type of reset	RTC	Peripheral unit	CPU core	PMUINTREG
RTC reset	Reset	Reset	Cold reset	RTCRST=1
RSTSW reset	Active	Reset	Cold reset	RSTSW=1

(1) RTC Reset

When the RTCRST* signal is asserted, the PMU resets all peripheral units including the RTC unit by asserting the rtcrsib and rst_gab signals (internal) and resets the CPU core by asserting the cooldresetb and creset signals (internal).

Further, it sets the RTCRST bit of PMUINTREG to 1. After the CPU has been restarted, the RTCRST bit must be checked and cleared by software.

(2) RSTSW Reset

When the RSTSW* signal is asserted, the PMU resets all peripheral units excluding the RTC and PMU by asserting the rst_gab signal (internal) and resets the CPU core by asserting the cooldresetb and creset signals (internal).

Further, it sets the RSTSW bit of PMUINTREG to 1. After the CPU has been restarted, the RSTSW bit must be checked and cleared by software.

15.1.2 Shutdown Control

A list of the states of the RTC, peripheral unit, CPU core, and the bit of PMUINTREG to be set during a shutdown is shown below.

Table 15-2. Types of Shutdown and Processor Status

Type of shutdown	RTC	Peripheral unit	CPU core	PMUINTREG
HAL timer shutdown	Active	Reset	Cold reset	HALTIMERRST =1
Deadman's SW shutdown	Active	Reset	Cold reset	TIMOUTRST=1
Hibernate shutdown	Active	Reset	Cold reset	-
Battery runout shutdown	Active	Reset	Cold reset	TIMOUTRST=1
Battery lock release shutdown	Active	Reset	Cold reset	-

(1) HAL Timer Shutdown

Software is required to write 1 to the HALTMERRST bit of PMUINTREG within approx. 4 seconds after the CPU has been restarted (the state where the shutdown state or Hibernate mode state have shifted to the Fullspeed mode) and reset the HALTimer.

If the HAL timer is not reset within approx. 4 seconds after the CPU has been restarted, the PMU resets all peripheral units excluding the RTC and PMU by asserting the rst_gab signal (internal) and resets the CPU core by asserting the cooldresetb and creset signals (internal).

Further, it sets the TIMOUTRST bit of PMUINTREG to 1. After the CPU has been restarted, the TIMOUTRST bit must be checked and cleared by software.

(2) Deadman's SW Shutdown

When the Deadman's SW shutdown function has been enabled, software is required to write 1 to the DSWCLR bit of DSUCLRREG at each set time to clear the Deadman's SW counter (Refer to Chapter 17 for details).

If the Deadman's SW counter is not cleared within the set time, the PMU resets all peripheral units excluding the RTC and PMU by asserting the rst_gab signal (internal) and resets the CPU core by asserting the cooldresetb and creset signals (internal).

Further, it sets the DMSRST bit of PMUINTREG to 1. After the CPU has been restarted, the TIMOUTRST bit must be checked and cleared by software.

(3) Software Shutdown

When the HIBERNATE instruction is executed, the PMU checks for the interrupts that are now on pending. When there are no interrupts on pending, it stops the CPU clock by asserting the cclockstopen signal. Further, it resets all peripheral units excluding the RTC and PMU by asserting the rst_gab signal (internal).

The contents of the PMU register are left unchanged.

15.1.3 Power-on Control

The factors to start the CPU (to set the state where the shutdown state or Hibernate mode state have shifted to the Fullspeed mode) are called starting factors and there are three types of starting factors: power-switch interrupt, DCD interrupt, and alarm interrupt.

On the other hand, there are two types of factors to hinder the starting of the CPU: detection of battery runout and detection of battery lock interrupt. However, when the starting of the CPU is hindered by the detection of battery lock interrupt, the operation of the CPU becomes unstable, so be sure to set the GPIO[9] (BATTLOCK) terminal to the High state (Lock state) when the CPU is in the Hibernate state or shutdown state (MPOWER=0).

(1) Starting by a Power-Switch Interrupt

When the POWER signal is asserted, the PMU notices other units that it is going to start the CPU by asserting the POWERON signal. After asserting the POWERON signal and checking the BATTINH signal, the PMU deasserts the POWERON signal.

If the BATTINH signal is High ("1"), the PMU releases the reset of peripheral units by deasserting the rst_gab signal (internal) and starts the CPU core by staring the cold reset sequence.

If the BATTINH signal is Low ("0"), the PMU shuts down again by setting the BATTINH bit of PMUINTREG to 1. After the CPU has been restarted, the BATTINH bit must be checked and cleared by software.

Figure 15-1. Starting by a Power-Switch Interrupt (BATTINH=1)

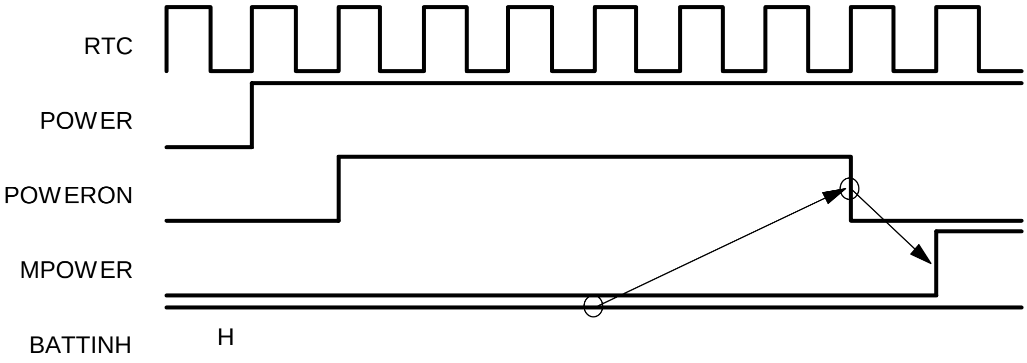
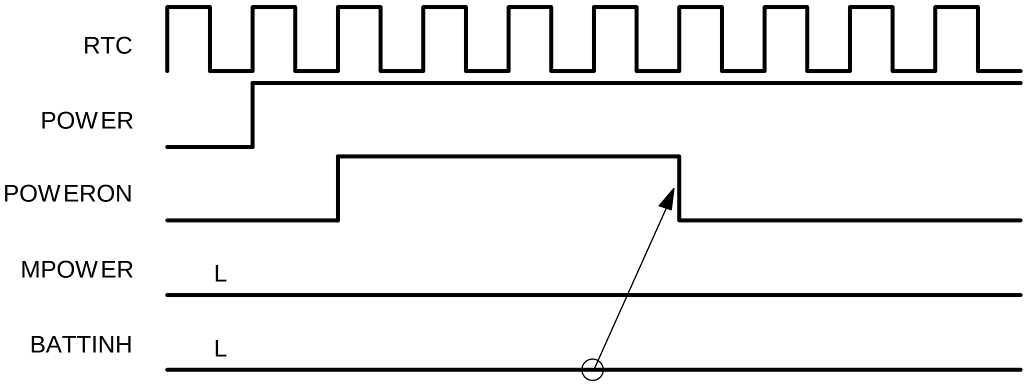


Figure 15-2. Starting by a Power-Switch Interrupt (BATTINH=0)



(2) Starting by a DCD interrupt

When the DCD signal is asserted, the PMU notices other units that it is going to start the CPU by asserting the POWERON signal. After asserting the POWERON signal and checking the BATTINH signal, the PMU deasserts the POWERON signal.

If the BATTINH signal is High ("1"), the PMU releases the reset of peripheral units by deasserting the rst_gab signal (internal) and starts the CPU core by staring the cold reset sequence.

If the BATTINH signal is Low ("0"), the PMU shuts down again by setting the BATTINH bit of PMUINTREG to 1. After the CPU has been restarted, the BATTINH bit must be checked and cleared by software.

The DCDST bit of PMUINTREG of the PMU does not indicate the presence or absence of a DCD interrupt but reflects the current state of the DCD terminal.

Figure 15-3. Starting by a DCD Interrupt (BATTINH=1)

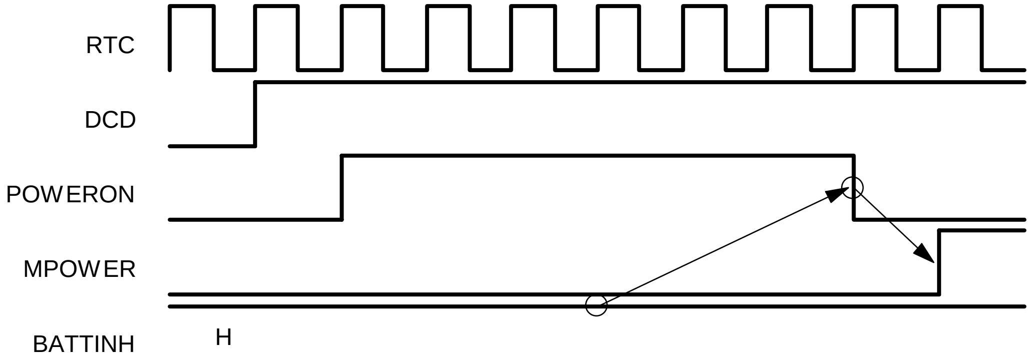
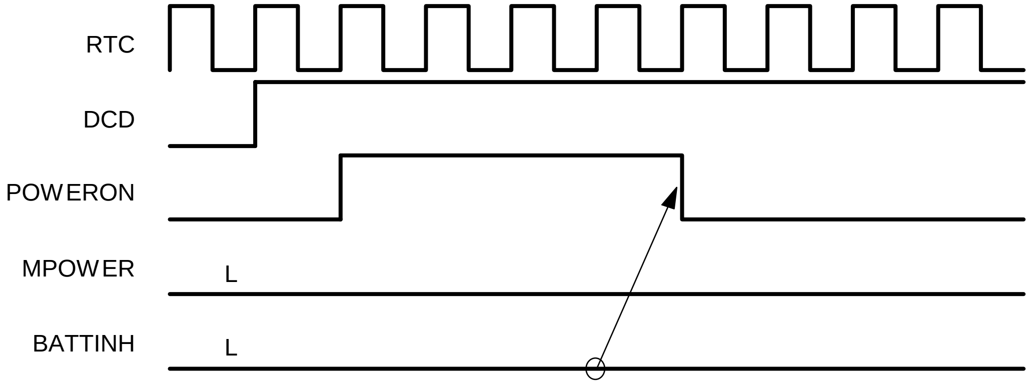


Figure 15-4. Starting by a DCD Interrupt (BATTINH=0)



(3) Starting by an Alarm Interrupt

When the interrupt signal by the alarm timer (alarm_intr) signal is asserted, the PMU notices other units that it is going to start the CPU by asserting the POWERON signal. After asserting the POWERON signal and checking the BATTINH signal, the PMU deasserts the POWERON signal.

If the BATTINH signal is High ("1"), the PMU releases the reset of peripheral units by deasserting the rst_gab signal (internal) and starts the CPU core by staring the cold reset sequence.

If the BATTINH signal is Low ("0"), the PMU shuts down again by setting the BATTINH bit of PMUINTREG to 1. After the CPU has been restarted, the BATTINH bit must be checked and cleared by software.

Figure 15-5. Starting by an Alarm Interrupt (BATTINH=1)

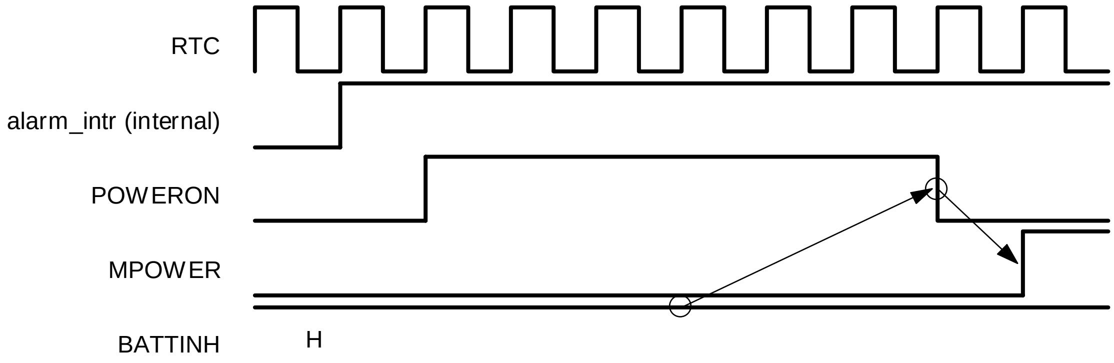
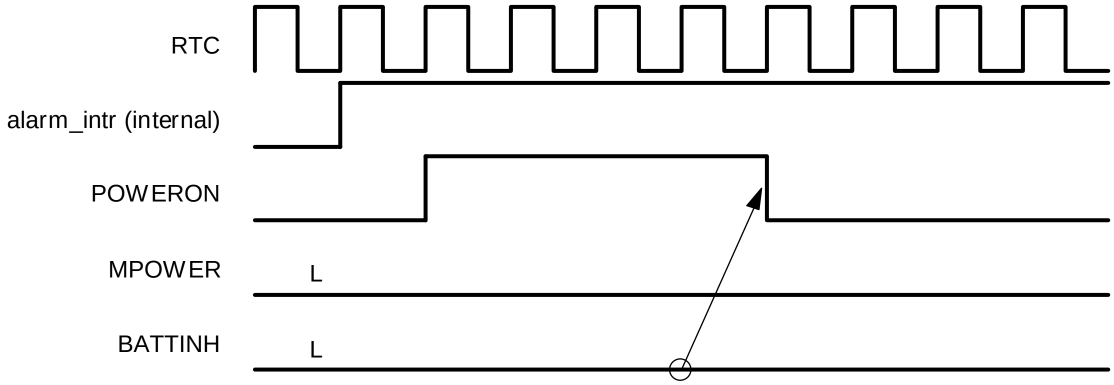


Figure 15-6. Starting by an Alarm Interrupt (BATTINH=0)



15.1.4 Power Mode

The VR4101 supports the following four power modes.

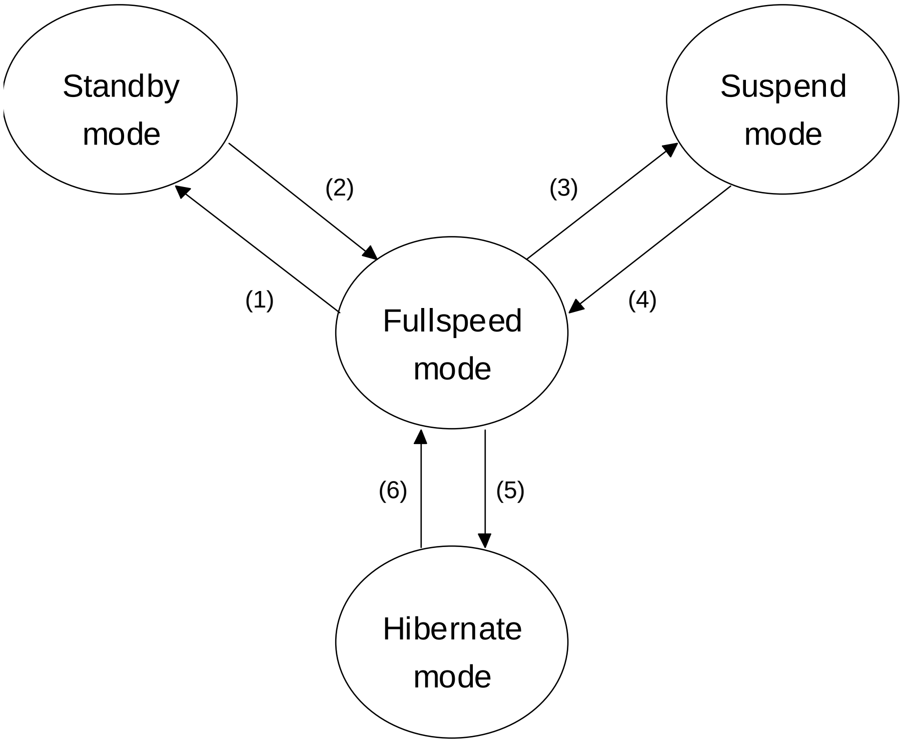
- ✧ Fullspeed mode
- ✧ Standby mode
- ✧ Suspend mode
- ✧ Hibernate mode

Figure 15-7 illustrates the transition between the different power modes.

To set Standby, Suspend, or Hibernate mode from Fullspeed mode, execute a STANDBY, SUSPEND, or HIBERNATE instruction. To set Fullspeed mode from Standby, Suspend, or Hibernate mode, generate an interrupt or perform any reset.

Table 15-3 outlines the power modes.

Figure 15-7. Power Mode Status Transition



(1)	(2)	(3)	(4)	(5)	(6)
STANDBY instruction & pipeline flash & SysAD idle & PClock high	All interrupts	SUSPEND instruction & pipeline flash & SysAD idle & PClock high & TClock high & DRAM self refresh	BatteryInt POWERSW RTCRST Alarm KeyTouch PenTouch BatteryLock CardLock DCD	HIBERNATE instruction & pipeline flash & SysAD idle & PClock high & TClock high & MasterOut high & DRAM self refresh	POWERSW Alarm DCD

Table 15-3. Power Mode

Mode	Internal peripheral unit				Power dissipation [Note1] (33 MHz, 3.3 V, typ.)
	RTC	ICU	DCU	others	
Fullspeed	On	On	On	Selectable [Note2]	200 mW
Standby	On	On	On	Selectable [Note2]	100 mW
Suspend	On	On	Off	Off	13 mW
Hibernate	On	Off	Off	Off	165 µW
Off	Off	Off	Off	Off	0 W

Notes **1.** Target value

2. See Chapter 13 for details.

(1) Fullspeed Mode

In Fullspeed mode, all internal clocks and the system interface clock operate. In this mode, all the functions of the VR4101 can be executed.

(2) Standby Mode

In Standby mode, all internal clocks, other than those provided to the internal peripheral units and the internal timer/interrupt unit of the CPU core, are fixed to high level.

To switch to Standby mode from Fullspeed mode, first execute the STANDBY instruction. The VR4101 waits until the SysAD bus (internal) enters idle status after the completion of the WB stage of the STANDBY instruction. Then, the internal clock is shut down, and the pipeline stops. PLL, timer/interrupt clock, internal bus clocks (TClock, MasterOut), and RTC continue to operate.

In Standby mode, the processor returns to Fullspeed mode when an interrupt occurs. At this time, the contents of bits indicating the states of terminals in the I/O registers are undefined. The contents of other fields are retained.

(3) Suspend Mode

In Suspend mode, all internal clocks (including TClock) other than those supplied to the RTC/ICU/PMU internal peripheral units and the internal timer/interrupt unit of the CPU core are fixed to high level.

To switch to Suspend mode from Fullspeed mode, first execute the SUSPEND instruction. The VR4101 waits until the SysAD bus (internal) enters idle status after the completion of the WB stage of the SUSPEND instruction, DRAM has entered self-refresh mode, and the MPOWER pin has been made inactive. Then, the internal clocks (including TClock) are shut down, and the pipeline stops. PLL, timer interrupt clock, MasterOut, and RTC continue to operate.

In Suspend mode, the processor returns to Fullspeed mode when an interrupt request from the peripheral units or any resets occur. At this time, the contents of bits indicating the states of terminals in the I/O registers are undefined. The contents of other fields are retained.

(4) Hibernate Mode

In Hibernate mode, all the clocks supplied to internal peripheral units other than RTC/ICU/PMU and to the CPU core are fixed to high level.

To switch to Hibernate mode from Fullspeed mode, first execute the HIBERNATE instruction. The VR4101 waits until the SysAD bus (internal) enters idle status after the completion of the WB stage of the HIBERNATE instruction, DRAM has entered self-refresh mode, and the MPOWER pin has been made inactive. Then, the internal clocks (including TClock and MasterOut) are shut down, and the pipeline stops. PLL also stops, but RTC continue to operate.

In Hibernate mode, the processor returns to Fullspeed mode when it is alarmed from the RTC, the power-on switch is pressed, or DCD pin is asserted. At this time, the contents of bits indicating the states of terminals in the I/O registers and caches in the CPU core are undefined. The contents of other fields are retained.

15.2 REGISTER SET

The following table lists the registers of the PMU.

Table 15-4. PMU Registers

Address	R/W	Register symbols	Function
0x0B00 00A0	R/W1C	PMUINTREG	PMU Interrupt/Status register
0x0B00 00A2	R/W	PMUCNTREG	PMU Control register

The function of each of these registers is explained in detail below.

15.2.1 PMUINTREG

Figure 15-8. PMUINTREG (0x0B00 00A0) (1/2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	DCDST	RTCINTR	BATTINH
R/W	R	R	R	R	R	R	R/W1C	R/W1C
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	BATT LOCK	CARD LOCK	TIMOUT RST	RTCRST	RSTSW	DMSRST	BATT INTR	POWER SWINTR
R/W	R/W	R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..11]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[10]	DCDST	DCD pin status 1: High level 0: Low level
D[9]	RTCINTR	Detects RTC alarm interrupt. 1: Detected 0: Not detected
D[8]	BATTINH	Detects low battery voltage on power up. 1: Detected 0: Not detected
D[7]	BATTLOCK	Detects battery lock interrupt. 1: Detected 0: Not detected
D[6]	CARDLOCK	Detects PCMCIA card lock. 1: Detected 0: Not detected
D[5]	TIMOUTRST	Detects HAL timer reset. 1: Detected 0: Not detected
D[4]	RTCRST	Detects RTC reset. 1: Detected 0: Not detected
D[3]	RSTSW	Detects reset SW interrupt. 1: Detected 0: Not detected

Figure 15-8. PMUINTREG (0x0B00 00A0) (2/2)

Bit position	Bit name	Function
D[2]	DMSRST	Detects Deadman’s switch interrupt. 1: Detected 0: Not detected
D[1]	BATTINTR	Detects low battery voltage interrupt during normal operation. 1: Detected 0: Not detected
D[0]	POWERSW INTR	Detects power switch interrupt. 1: Detected 0: Not detected

15.2.2 PMUCNTREG

Figure 15-9. PMUCNTREG (0x0B00 00A2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	STANDBY	Reserved	Reserved	Reserved	Reserved	HALTIME RRST	Reserved	Reserved
R/W	R/W	R	R	R	R	R/W	R	R
Initial value	0	0	0	0	0	0	1	0

Bit position	Bit name	Function
D[15..8]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[7]	STANDBY	Sets Standby mode. This setting is performed only for software, and does not affect hardware in any way. 1: Standby mode 0: Normal mode
D[6..3]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[2]	HALTIMERRST	Resets HAL timer. 1: Reset 0: Set
D[1]	Reserved	Reserved for future use. Write 1 to this bit. 1 is returned when this bit is read.
D[0]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.

This register performs the setting of CPU shutdown and management of the whole system.

Be sure to reset the HALTIMERRST bit within approx. 4 seconds of applying power. By the reset this HALTIMERRST bit, the VR4101 can recognize that itself has started normally. If the HALTIMERRST bit is not reset within approx. 4 seconds of applying power, the VR4101 assumes that the program cannot be executed normally (that the program may have hung) and automatically shuts down.

[MEMO]

CHAPTER 16 RTC (REALTIME CLOCK UNIT)

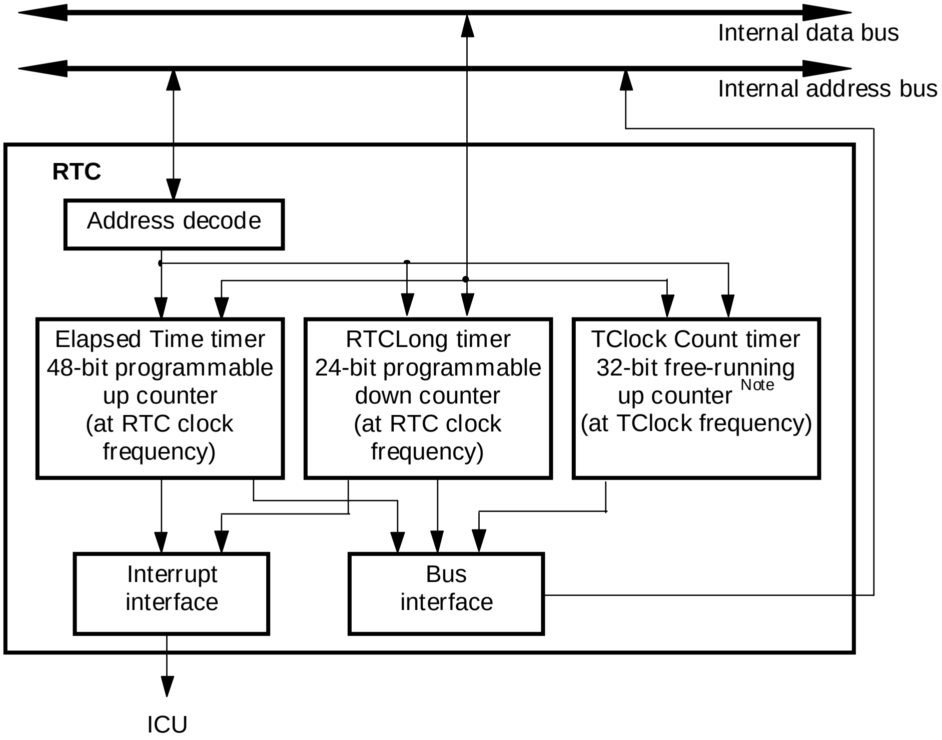
This chapter explains the operation of the RTC and how to set the registers of the RTC.

16.1 GENERAL

The RTC provides the following three timers:

- ✧ RTCLong ....24-bit programmable counter that counts at 32.768-kHz. Generates a cyclic interrupt at intervals of up to 512 seconds.
- ✧ ElapsedTime ..48-bit up counter that counts at 32.768-kHz. Counts up to approximately 272 years and then returns to zero. Can generate an interrupt at a specific time by comparing the ElapsedTime (ETIMELREG, ETIMRMREG, ETIMEHREG) with 48-bit alarm time register (ECMPHREG, EMPLREG, ECMPMREG).
- ✧ TClockCount ..Free-running counter that counts up at the TClock frequency. Used for performance evaluation.

Figure 16-1. Functional Block Diagram of the RTC



Note   The MSB bit is a mask bit.

16.2 REGISTER SET

The following table lists the details of each register.

Table 16-1. RTC Registers

Address	R/W	Register symbol	Function
0x0B00 00C4	R/W	ETIMELREG	Elapsed Time L register
0x0B00 00C6	R/W	ETIMEMREG	Elapsed Time M register
0x0B00 00C8	R/W	ETIMEHREG	Elapsed Time H register
0x0B00 00CA	R/W	ECMPHREG	Elapsed Compare H register
0x0B00 00CC	R/W	ECMPLREG	Elapsed Compare L register
0x0B00 00CE	R/W	ECMPMREG	Elapsed Compare M register
0x0B00 00D0	R/W	RTCLLREG	RTC Long L register
0x0B00 00D2	R/W	RTCLHREG	RTC Long H register
0x0B00 00D4	R	RTCLCNTLREG	RTC Long Count L register
0x0B00 00D6	R	RTCLCNTHREG	RTC Long Count H register
0x0B00 00D8	R/W	TCLKCNTLREG	TCLK Count L register
0x0B00 00DA	R/W	TCLKCNTHREG	TCLK Count H register
0x0B00 00DC	R/W1C	RTCINTREG	RTC Interrupt register

The function of each of these registers is explained in detail below.

16.2.1 ETIMELREG, ETIMEMREG, ETIMEHREG

These registers are used to set and indicate count value of the ElapsedTime timer.

The ElapsedTime timer is a 48-bit counter that counts out at 30 μs cycle (32.768 KHz) and can count up to approximately 272 years.

Initialization in terms of hardware is effected only on the RTCRST* terminal.

Figure 16-2. ETIMELREG (0x0B00 00C4)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	ETIMEL [15]	ETIMEL [14]	ETIMEL [13]	ETIMEL [12]	ETIMEL [11]	ETIMEL [10]	ETIMEL [9]	ETIMEL [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	ETIMEL [7]	ETIMEL [6]	ETIMEL [5]	ETIMEL [4]	ETIMEL [3]	ETIMEL [2]	ETIMEL [1]	ETIMEL [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	ETIMEL[15..0]	Bits 15 through 0 of ElapsedTime timer

Figure 16-3. ETIMEMREG (0x0B00 00C6)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	ETIMEM [31]	ETIMEM [30]	ETIMEM [29]	ETIMEM [28]	ETIMEM [27]	ETIMEM [26]	ETIMEM [25]	ETIMEM [24]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	ETIMEM [23]	ETIMEM [22]	ETIMEM [21]	ETIMEM [20]	ETIMEM [19]	ETIMEM [18]	ETIMEM [17]	ETIMEM [16]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	ETIMEM[31..16]	Bits 31 through 16 of ElapsedTime timer

Figure 16-4. ETIMEHREG (0x0B00 00C8)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	ETIMEH [47]	ETIMEH [46]	ETIMEH [45]	ETIMEH [44]	ETIMEH [43]	ETIMEH [42]	ETIMEH [41]	ETIMEH [40]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	ETIMEH [39]	ETIMEH [38]	ETIMEH [37]	ETIMEH [36]	ETIMEH [35]	ETIMEH [34]	ETIMEH [33]	ETIMEH [32]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	ETIMEH[47..32]	Bits 47 through 32 of ElapsedTime timer

16.2.2 ECMPHREG, ECMPLREG, ECMPMREG

These registers are used to set a value compared with the ElapsedTime timer. Comparison is started at the rising edge of the second RTC clock after the setting of these registers. An interrupt is generated when the contents of these registers matches those of the ElapsedTime timer.

Figure 16-5. ECMPHREG (0x0B00 00CA)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	ECMPH [47]	ECMPH [46]	ECMPH [45]	ECMPH [44]	ECMPH [43]	ECMPH [42]	ECMPH [41]	ECMPH [40]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	ECMPH [39]	ECMPH [38]	ECMPH [37]	ECMPH [36]	ECMPH [35]	ECMPH [34]	ECMPH [33]	ETIMEH [32]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	ECMPH[47..32]	Value to be compared with bits 47 through 32 of ElapsedTime timer

Figure 16-6. ECMPLREG (0x0B00 00CC)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	ECMPL [15]	ECMPL [14]	ECMPL [13]	ECMPL [12]	ECMPL [11]	ECMPL [10]	ECMPL [9]	ECMPL [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	ECMPL [7]	ECMPL [6]	ECMPL [5]	ECMPL [4]	ECMPL [3]	ECMPL [2]	ECMPL [1]	ECMPL [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	ECMPL[15..0]	Value to be compared with bits 15 through 0 of ElapsedTime timer

Figure 16-7. ECMPMREG (0x0B00 00CE)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	ECMPM [31]	ECMPM [30]	ECMPM [29]	ECMPM [28]	ECMPM [27]	ECMPM [26]	ECMPM [25]	ECMPM [24]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	ECMPM [23]	ECMPM [22]	ECMPM [21]	ECMPM [20]	ECMPM [19]	ECMPM [18]	ECMPM [17]	ECMPM [16]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	ECMPM[31..16]	Value to be compared with bits 31 through 16 of ElapsedTime timer

16.2.3 RTCLLREG, RTCLHREG

Initialization in terms of hardware is effected only on the RTCRST* terminal.

RTCLLREG and RTCLHREG are the registers to set the cycle of RTCLong timer. By performing storing operation on both registers of RTCLLREG and RTCLHREG (at TClock cycle), the set cycle of the RTCLong timer is changed. Storing to either register does not effect the change in the set cycle. In this case, set cycle maintains the former value. The write flags for these lower-order and higher-order bits are cleared when both have become 1 or they are reset.

For example, when the "cycle" is "m," countdown is repeated as "m" → "m-1" → ... → "2" → "1" (an interrupt occurs here) → "m" → ... "1."

The RTCLong timer is a 24-bit programmable counter that counts at 30 μs cycle (32.768 kHz), and is used for generating up to 512 sec of periodical interrupts.

In the current implement, the RTCLong timer stops when 0 is set as the "cycle." The minimum value that can be set is 4. Be sure to set these registers to 4 or greater value.

Figure 16-8. RTCLLREG (0x0B00 00D0)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	RTCLPL [15]	RTCLPL [14]	RTCLPL [13]	RTCLPL [12]	RTCLPL [11]	RTCLPL [10]	RTCLPL [9]	RTCLPL [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	RTCLPL [7]	RTCLPL [6]	RTCLPL [5]	RTCLPL [4]	RTCLPL [3]	RTCLPL [2]	RTCLPL [1]	RTCLPL [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	RTCLPL[15..0]	Bits 15 through 0 of RTCLong timer interrupt cycle

Figure 16-9. RTCLHREG (0x0B00 00D2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	RTCLPH [23]	RTCLPH [22]	RTCLPH [21]	RTCLPH [20]	RTCLPH [19]	RTCLPH [18]	RTCLPH [17]	RTCLPH [16]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..8]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[7..0]	RTCLPH[23..16]	Bits 23 through 16 of RTCLong timer interrupt cycle

16.2.4 RTCLCNTLREG, RTCLCNTHREG

Initialization in terms of hardware is effected only on the RTCRST* terminal.

RTCLCNTLREG and RTCLCNTHREG operate as a 24-bit counter that perform countdown based on the cycle set on the RTCLLREG and RTCLHREG. Read is performed in two sessions because of the internal bus of 16-bit type. In this case, erroneous data may be returned if there is any carry of a digit.

An interrupt occurs at the cycle that follows the cycle which these registers indicate "1." At the same time, these registers take the value of RTCLLREG and RTCLHREG, and then continue countdown.

Figure 16-10. RTCLCNTLREG (0x0B00 00D4)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	RTCLCL [15]	RTCLCL [14]	RTCLCL [13]	RTCLCL [12]	RTCLCL [11]	RTCLCL [10]	RTCLCL [9]	RTCLCL [8]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	RTCLCL [7]	RTCLCL [6]	RTCLCL [5]	RTCLCL [4]	RTCLCL [3]	RTCLCL [2]	RTCLCL [1]	RTCLCL [0]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	RTCLCL[15..0]	Bits 15 through 0 of RTCLong timer

Figure 16-11. RTCLCNTHREG (0x0B00 00D6)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	RTCLCH [23]	RTCLCH [22]	RTCLCH [21]	RTCLCH [20]	RTCLCH [19]	RTCLCH [18]	RTCLCH [17]	RTCLCH [16]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..8]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[7..0]	RTCLCH[23..16]	Bits 23 through 16 of RTCLong timer

16.2.5 TCLKCNTLREG, TCLKCNTHREG

These registers are used to set the count value of TClock Count timer.

The TClock Count timer is a 32-bit register that performs count-up based on TClock. A write to this register is enabled only for the diagnostic purpose.

Bit 31 is not the value to written or read to and from the counter, but disables timer operation with "0" (reset/stop) or enables with "1."

Counter operation is disabled by a reset and the counter is initialized.

Figure 16-12. TCLKCNTLREG (0x0B00 00D8)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	TCLKCL [15]	TCLKCL [14]	TCLKCL [13]	TCLKCL [12]	TCLKCL [11]	TCLKCL [10]	TCLKCL [9]	TCLKCL [8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	TCLKCL [7]	TCLKCL [6]	TCLKCL [5]	TCLKCL [4]	TCLKCL [3]	TCLKCL [2]	TCLKCL [1]	TCLKCL [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..0]	TCLKCL[15..0]	Bits 15 through 0 of TClock counter

Figure 16-13. TCLKCNTHREG (0x0B00 00DA)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	TCLKEN	TCLKC H [30]	TCLKCH [29]	TCLKCH [28]	TCLKCH [27]	TCLKCH [26]	TCLKCH [25]	TCLKCH [24]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	TCLKCH [23]	TCLKCH [22]	TCLKCH [21]	TCLKCH [20]	TCLKCH [19]	TCLKCH [18]	TCLKCH [17]	TCLKCH [16]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15]	TCLKEN	Enables TClock Counter operation 1: Enabled 0: Disabled
D[14..0]	TCLKCH[30..16]	Bits 30 through 16 of TClock Counter

16.2.6 RTCINTREG

This register indicates the occurrence of interrupts generated by the ElapsedTime timer and the RTCLong timer.

To write 1 to the corresponding bit of this register can also generate an interrupt.

Figure 16-14. RTCINTREG (0x0B00 00DC)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	RTCINTR1	RTCINTR0
R/W	R	R	R	R	R	R	R/W1C	R/W1C
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..2]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[1]	RTCINTR1	ElapsedTime interrupt 1: ElapsedTime interrupt occurs 0: Normal
D[0]	RTCINTR0	RTCLong interrupt 1: RTCLong interrupt occurs 0: Normal

CHAPTER 17 DSU (DEADMAN'S SW UNIT)

This chapter explains the operation of the DSU and how to set the registers of the DSU.

17.1 GENERAL

Should the VR4101 hang up, this is automatically detected by the DSU, which resets the VR4101 to minimize the hang-up time. By minimizing the hang-up time, the destruction of data caused by the software hanging up can be minimized.

17.2 REGISTER SET

The following table lists the registers of the DSU.

Table 17-1. DSU Registers

Address	R/W	Register symbols	Function
0x0B00 00E0	R/W	DSUCNTREG	DSU Control register
0x0B00 00E2	R/W	DSUSETREG	DSU Dead Time Set register
0x0B00 00E4	W1C	DSUCLRREG	DSU Clear register
0x0B00 00E6	R/W	DSUTIMREG	DSU Elapsed Time register

The function of each of these registers is explained in detail below.

17.2.1 DSU Control Register

Figure 17-1. DSUCNTREG (0x0B00 00E0)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	DSWEN
R/W	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	DSWEN	Enables Deadman's switch function. 1: Enabled 0: Disabled

This register is used to enable the Deadman's switch function.

17.2.2 DSU Dead Time Setting Register

Figure 17-2. DSUSETREG (0x0B00 00E2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	DEDTIME [3]	DEDTIME [2]	DEDTIME [1]	DEDTIME [0]
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..4]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[3..0]	DEDTIME[3..0]	Sets Deadman’s switch cycle. 1111 15 seconds 1001 9 seconds 0011 3 seconds 1110 14 seconds 1000 8 seconds 0010 2 seconds 1101 13 seconds 0111 7 seconds 0001 1 second 1100 12 seconds 0110 6 seconds 0000 Reserved for future use. 1011 11 seconds 0101 5 seconds 1010 10 seconds 0100 4 seconds

This register sets the Deadman’s switch cycle.

The Deadman’s switch cycle can be set to between 1 and 15 seconds in units of 1 second. If, however, DEDTIME[3..0] are set to 0x0, the operation of the VR4101 will be undefined.

17.2.3 DSU Clear Register

Figure 17-3. DSUCLRREG (0x0B00 00E4)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	DSWCLR
R/W	R	R	R	R	R	R	R	W1C
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..1]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[0]	DSWCLR	Clears Deadman's switch. 1: Clears 0: Does not clear

This register clears the Deadman's switch counter. The software must set the DSWCLR bit of this register within the cycle set with the DSUSETREG. If this bit is not cleared within the cycle, the VR4101 is considered to be hanging up and is automatically reset.

17.2.4 DSU Elapsed Time Register

Figure 17-4. DSUTIMREG (0x0B00 00E6)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	Reserved	CRTTIME [3]	CRTTIME [2]	CRTTIME [1]	CRTTIME [0]
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..4]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[3..0]	CRTTIME[3..0]	Elapsed time of Deadman’s switch timer 1111 15 seconds 1001 9 seconds 0011 3 seconds 1110 14 seconds 1000 8 seconds 0010 2 seconds 1101 13 seconds 0111 7 seconds 0001 1 second 1100 12 seconds 0110 6 seconds 0000 Reserved for future use. 1011 11 seconds 0101 5 seconds 1010 10 seconds 0100 4 seconds

This register indicates the elapsed time of the current Deadman’s switch.

17.3 REGISTER SETTING FLOW

The register setting flow for the DSU is shown below.

1. Set the count-up value of the DSU (1 to 15 seconds).
   If the timer is not cleared by the CPU within this time, the CPU is reset.

   DSUDTMREG address : 0x0B00 00E2 data : 0x000n (n = 1 to F)

2. Enable the DSU.

   DSUCNTREG address : 0x0B00 00E0 data : 0x0001

3. Clear the timer with the time set in 1.

   DSUCLRREG address : 0x0B00 00E4 data : 0x0001

   For ordinary use, always 3. is repeated.
   To know the time elapsed until now, set as follows:

   DSITIMREG address : 0x0B00 00E6 read (4 bit)

4. Disable the DSU for the Suspend mode or Shutdown.

   DSUCNTREG address : 0x0B00 00E0 data : 0x0000

CHAPTER 18 GIU (GENERAL PURPOSE I/O UNIT)

This chapter explains the operation of the GIU and how to set the registers of the GIU.

18.1 GENERAL

The GIU controls GPIO[11..0] and DCD terminals. One of GPIO[11..0] is already assigned to a specific function, however GPIO terminals are used as a port which supports output and input. The other eleven GPIO and DCD terminals can be assigned to interrupt requests, and three types of interrupt triggers are selectable: changes in the input signal (rising or falling edge), a low level of the input signal, or a high level of the input signal. The following table lists the types of input buffers and clocks used to detect interrupt requests.

Table 18-1. Outline of GPIO Pins and DCD Pin

Pin name	Interrupt detection clock	Input buffer type
DCD Note1	MasterOut	---
GPIO[11]	TClock	Normal
GPIO[10]	MasterOut	Schmitt
GPIO[9] Note2	MasterOut	Schmitt
GPIO[8]	TClock	Normal
GPIO[7]	TClock	Normal
GPIO[6]	TClock	Normal
GPIO[5]	TClock	Normal
GPIO[4]	TClock	Normal
GPIO[3]	TClock	Normal
GPIO[2]	TClock	Normal
GPIO[1]	TClock	Normal
GPIO[0]	TClock	Normal

Note 1. DCD pin (input) is internally connected to bit 13 of the GPIO registers. GIU supports the function of DCD pin as an input only.

2. GPIO[9] pin must be assigned to the battery cover lock detection signal (BATTLOCK).

18.2 REGISTER SET

The following table lists the registers of the GIU.

Table 18-2. GIU Registers

Address	R/W	Register symbols	Function
0x0B00 0100	R/W	GOUTENREG	GPIO Output Enable register
0x0B00 0102	R/W	GPOTDATREG	GPIO Port Data register
0x0B00 0104	R/W1C	GINTSTREG	GPIO Interrupt Status register
0x0B00 0106	R/W	GINTENREG	GPIO Interrupt Enable register
0x0B00 0108	R/W	GCINTSREG	GPIO Change Point Interrupt register
0x0B00 010A	R/W	GLINTSREG	GPIO Interrupt Level Specified register

The function of each of these registers is explained in detail below.

18.2.1 GPIO Output Enable Register

Figure 18-1. GOUTENREG (0x0B00 0100)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	IOP[11]	IOP[10]	IOP[9]	IOP[8]
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IOP[7]	IOP[6]	IOP[5]	IOP[4]	IOP[3]	IOP[2]	IOP[1]	IOP[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IOP[11..0]	Sets input/output mode for GPIO[11..0] pins. 1: Output 0: Input (Hi-Z)

This register sets input/output mode for the GPIO[11..0] pins.

IOP[11..0] bits correspond to the input/output status of the GPIO[11..0] pins. When an IOP bit is set to 1, the corresponding GPIO pin is set to output mode, and outputs the value written to the corresponding IODATA of GIUDATAREG. When the IOP bit is cleared to 0, the corresponding GPIO pin enters the high-impedance state and is set to input mode.

18.2.2 GPIO Port Data Register

Figure 18-2. GPOTDATREG (0x0B00 0102)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	IODATA [13]	Reserved	IODATA [11]	IODATA [10]	IODATA [9]	IODATA [8]
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IODATA [7]	IODATA [6]	IODATA [5]	IODATA [4]	IODATA [3]	IODATA [2]	IODATA [1]	IODATA [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	IODATA[13]	DCD pin data 1: High 0: Low
D[12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IODATA[11..0]	GPIO[11..0] pins data 1: High 0: Low

This register is used to read/write the data of the DCD and GPIO[11..0] pins.

The IODATA[11..0] bits correspond to the data of the GPIO[11..0] pins, while the IODATA[13] bit corresponds to the data of the DCD pin. When the corresponding IOP bit of the GIUOUTENREG is set to 1, the value written to an IODATA bit is output to the corresponding GPIO pin. The set data is output to GPIO pins synchronously with the rising edge of TClock. The GPIO pin is not affected even if a value is written to the corresponding IODATA bit when the corresponding IOP bit is cleared to 0.

When an IODATA bit is read, the current status of the corresponding GPIO pin can be read.

18.2.3 GPIO Interrupt Status Register

Figure 18-3. GINTSTREG (0x0B00 0104)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	IOINTR [13]	Reserved	IOINTR [11]	IOINTR [10]	IOINTR [9]	IOINTR [8]
R/W	R	R	R/W1C	R	R/W1C	R/W1C	R/W1C	R/W1C
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IOINTR [7]	IOINTR [6]	IOINTR [5]	IOINTR [4]	IOINTR [3]	IOINTR [2]	IOINTR [1]	IOINTR [0]
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	IOINTR[13]	DCD pin interrupt 1: Occurred 0: Normal
D[12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IOINTR[11..0]	GPIO[11:0] pins interrupt 1: Occurred 0: Normal

This register indicates the status of the interrupt to DCD and GPIO[11..0] pins.

The IOINTR[11..0] bits correspond to the data of the GPIO[11..0] pins, while the IOINT[13] bit corresponds to the data of the DCD pin. If the corresponding IOP bit of the GIUINTENREG is set to 1 and if the signal input to the GPIO pin or DCD pin, whose interrupt is enabled, satisfies the condition specified by either GIUINTSREG or GIUINTLREG, the corresponding IOINTR bit is set to 1.

18.2.4 GPIO Interrupt Enable Register

Figure 18-4. GINTENREG (0x0B00 0106)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	IOINTR EN[13]	Reserved	IOINTR EN[11]	IOINTR EN[10]	IOINTR EN[9]	IOINTR EN[8]
R/W	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IOINTR EN[7]	IOINTR EN[6]	IOINTR EN[5]	IOINTR EN[4]	IOINTR EN[3]	IOINTR EN[2]	IOINTR EN[1]	IOINTR EN[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	IOINTREN[13]	Enables DCD pin interrupt. 1: Enabled 0: Disabled
D[12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IOINTREN[11..0]	Enables GPIO[11..0] pins interrupt. 1: Enabled 0: Disabled

This register enables the interrupt to the DCD and GPIO[11..0] pins.

The IOINTREN[11..0] bits correspond to the data of the GPIO[11..0] pins, while the IOINTREN[13] bit correspond to the data of the DCD pin. When the corresponding IOINTREN bit is set to 1, the interrupt to the corresponding GPIO pin or DCD pin is enabled. However, the interrupt occurs even if the GPIO pin is set to output mode by the GIUOUTENREG provided the output data of the GPIO pin satisfies the condition specified by either GIUINTSREG or GIUINTLREG. Therefore, clear the IOINTREN bit, corresponding to the pin corresponding to the GPIO pin set to output mode by the GIUOUTENREG, to 0 to disable the interrupt.

18.2.5 GPIO Change Point Interrupt Register

Figure 18-5. GCINTSREG (0x0B00 0108)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	IOINTR TYP[13]	Reserved	IOINTR TYP[11]	IOINTR TYP[10]	IOINTR TYP[9]	IOINTR TYP[8]
R/W	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IOINTR TYP[7]	IOINTR TYP[6]	IOINTR TYP[5]	IOINTR TYP[4]	IOINTR TYP[3]	IOINTR TYP[2]	IOINTR TYP[1]	IOINTR TYP[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	IOINTRTYP[13]	Sets DCD pin interrupt type. 1: Edge 0: Level
D[12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IOINTRTYP[11..0]	Sets GPIO[11..0] pin interrupt type. 1: Edge 0: Level

This register sets the types of the interrupts to the DCD and GPIO[11..0] pins.

The IOINTRTYP[11..0] bits correspond to the data of the GPIO[11..0] pins, and the IOINTRTYP[13] bit corresponds to the data of the DCD pin. If the corresponding IOINTRTYP bit is set to 1, the interrupt to the corresponding GPIO or DCD pin is latched by a rising or falling edge (i.e., the interrupt occurs when the corresponding pin goes high or low). When the IOINTRTYP bit is cleared to 0, the corresponding interrupt is latched by the signal level. Whether the interrupt is latched by a low or high level is specified by setting the corresponding IOINTLVL bit of the GLINTSREG.

18.2.6 Interrupt Level Identifying Register

Figure 18-6. GLINTSREG (0x0B00 010A)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	IOINTR LVL[13]	Reserved	IOINTR LVL[11]	IOINTR LVL[10]	IOINTR LVL[9]	IOINTR LVL[8]
R/W	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	IOINTR LVL[7]	IOINTR LVL[6]	IOINTR LVL[5]	IOINTR LVL[4]	IOINTR LVL[3]	IOINTR LVL[2]	IOINTR LVL[1]	IOINTR LVL[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..14]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[13]	IOINTRLVL [13]	Sets DCD pin level interrupt. 1: High active 0: Low active
D[12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	IOINTRLVL [11..0]	Sets GPIO[11..0] pins level interrupt. 1: High active 0: Low active

This register sets the types of the level interrupts to the DCD and GPIO[11..0] pins.

The IOINTRLVL[11..0] bits correspond to the data of the GPIO[11..0] pins, while the IOINTRLVL[13] bit corresponds to the data of the DCD pin. If the corresponding IOINTRLVL bit is set to 1, the interrupt to the corresponding GPIO or DCD pin is latched by a high level (High active). If the IOINTRLVL bit is cleared to 0, the interrupt is latched by a low level (Low active).

18.3 REGISTER SETTING FLOW

The setting flow for the GPIO is as shown below.

(1) Example of setting conditions

• Output:	GPIO[11:9]
• Input:	GPIO[8:0]
Interrupt change:	GPIO[8:6]
Low level:	GPIO[5:3]
High level:	GPIO[2:0]

(2) Setting flow

1. HAL Timer clear (PMU)
   ADD 0x0B00 00A2
   DAT 0x0005

2. Clock supply to the GIU
   ADD 0x0B00 0060
   DAT 0x0010

3. Enabling interrupts from the GPIO pins (ICU)

   msk_sysreg
   ADD 0x0B00 008C
   DAT 0x0300

   msk_giureg
   ADD 0x0B00 0094
   DAT 0x07FC

4. Setting GIU registers

   GOUTENREG (Setting input/output)
   ADD 0x0B00 0100
   DAT 0x1E00

   GPOTDATREG (Setting the output value)
   ADD 0x0B00 0100
   DAT 0x1E00

   GLINTSREG (Setting the interrupt level)
   ADD 0x0B00 010A
   DAT 0x0007

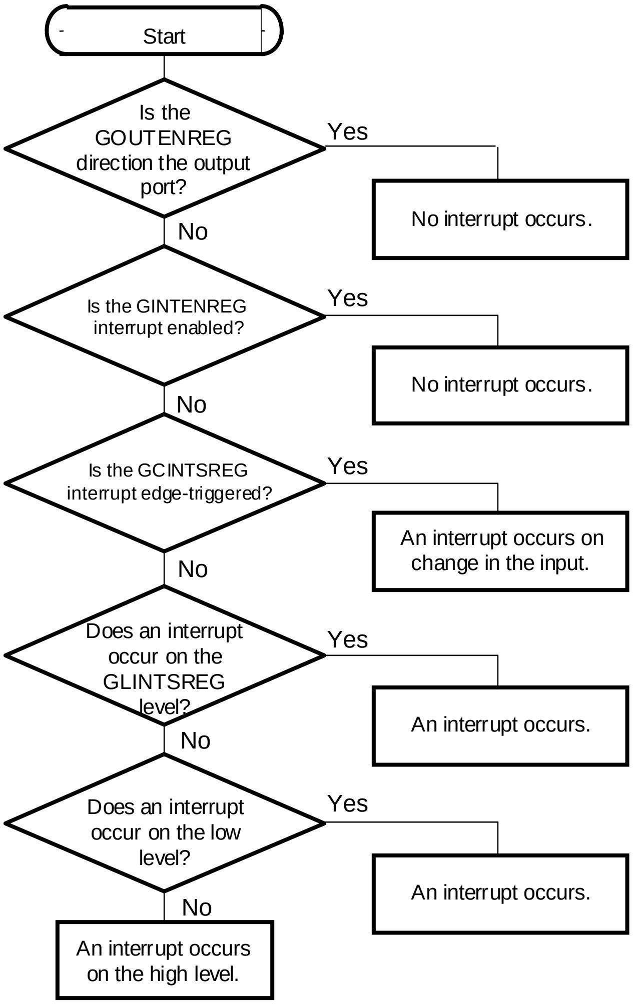
   GCINTSREG (Setting the interrupt change)
   ADD 0x0B00 0108
   DAT 0x01C0

   GINTENREG (Enabling interrupts)
   ADD 0x0B00 0106
   DAT 0x01FF

18.4 INTERRUPT FROM GPIO PINS

The following figure shows the flow chart of occurrence of an interrupt from GPIO pins.

Figure 18-7. Flow Chart of the Occurrence of an Interrupt


Start
Is the GOUTENREG direction the output port?
Yes
No interrupt occurs.
No
Is the GINTENREG interrupt enabled?
Yes
No interrupt occurs.
No
Is the GCINTSREG interrupt edge-triggered?
Yes
An interrupt occurs on change in the input.
No
Does an interrupt occur on the GLINTSREG level?
Yes
An interrupt occurs.
No
Does an interrupt occur on the low level?
Yes
An interrupt occurs.
No
An interrupt occurs on the high level.

18.5 FUNCTIONS TO ACHIEVE LOW POWER CONSUMPTION

This unit has the following functions to achieve low power consumption:

- Operation clock to the GPIO can be masked (enabled by setting on the register of the CMU).
- Because the following three types of interrupts are detected by the MasterOut (to which clock is always supplied) if TClock (can be masked) is not supplied to the GPIO or the GPIO is shut down, they can be generated at any time.

  - DCD interrupt (SIU)
  - GPIO[10:9] interrupts

CHAPTER 19 PIU (TOUCH PANEL INTERFACE UNIT)

This chapter explains the operation of the PIU and how to set the registers of the PIU.

19.1 GENERAL

The PIU detects the X and Y coordinates of the point where the pen is touched to the panel by using an external A/D converter. As a secondary function, it also measures the battery voltage. As the external A/D converter, the TI TLV1543C (conversion accuracy: 10 bits) and TLC2543C (conversion accuracy: 12 bits) are supported.

The functions of the PIU such as the detection of the X and Y coordinates on the panel and the measurement of battery voltage are implemented as follows:

Hardware functions: • Control of the external circuit

• Acceptance of coordinate/battery voltage data and data transfer.

Software functions: • Processing of coordinate/battery voltage data based on the data sampled by hardware

Features of the hardware portion of the PIU are as follows:

- I/F dedicated to touch panel based on 4-terminal-type resistance films
- I/F dedicated to two types of A/D converters (TC2543C and TLV1543C made by TI, Inc.)
- Main/sub battery voltage detection
- Control of the A/D converter and external circuit by arbitrary setting
- X and Y coordinate data and pen pressure data sampling
- Variable interval of coordinate data sampling
- Variable clock cycle for the A/D converter
- Generation of interrupts by pen touch
- DMA channel dedicated for the PIU
- Auto/manual is selectable for coordinate data sampling start/stop control

19.1.1 Block Diagram

Figure 19-1. Block Diagram of an Example of the Configuration of an External Circuit

Main battery Sub battery

◆ Touch panel

The touch panel has a total of four terminals at both ends of each of the X-directional and Y-directional resistance films; the resistance between the two film is high when the pen is not contacting, and is low when the pen is contacting. The resistance across the resistance film is about 1 k-ohms and the Y coordinate can be obtained by measuring the voltage across the terminals of the X-directional resistance film while applying voltage access the Y-directional resistance film. The X coordinate can also be obtained in the similar manner. Further, to enhance the accuracy of the detection of coordinates, measurement should be performed by changing the direction of the voltage to be applied to the resistance film. X and Y coordinate data can be obtained by performing a total of four voltage measurements.

Figure 19-2. Equalized Circuit for Detecting Coordinates

(a) for Y coordinates

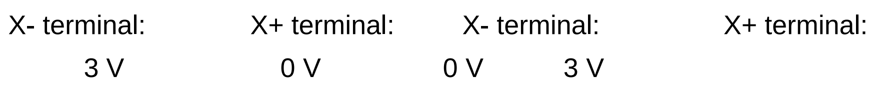
Y+ terminal: 3 V Y+ terminal: 0 V

X+ terminal X+ terminal

Y- terminal: 0 V Y- terminal: 3 V

(b) for X coordinates

Y+ terminal Y+ terminal



- ◆ Driver

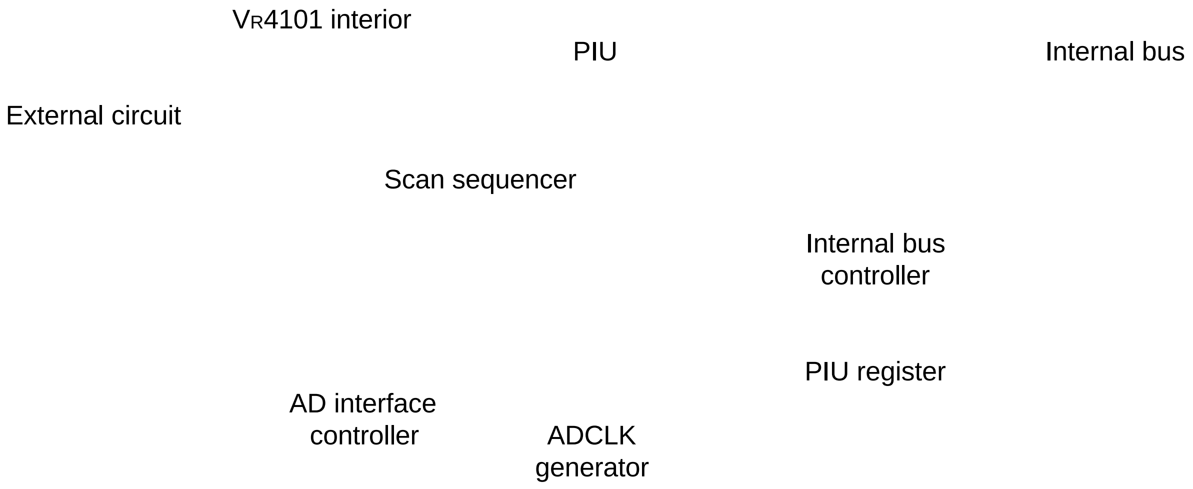
  This is the driver to apply voltage to the touch panel. This is controlled by PENCONT[4..0].

- ◆ A/D converter

  The TLV1543C (conversion accuracy: 10 bits) and TLC2543C (conversion accuracy: 12 bits) are applicable. For details about connection of an A/D converter, refer to VR4101 Application Note which is separately available.

  All controls of the A/D converter are performed by the PIU.

Figure 19-3. Block Diagram of the PIU Interior



The PIU is composed of the four blocks of internal bus controller, scan sequencer, ADCLK generator, and ADI/F controller.

- ◆ Internal bus controller

  The internal bus controller performs the control of internal bus, DMA, PIU register, interrupt, and serial-parallel conversion of the data from the A/D converter.

- Scan sequencer

  The scan sequencer performs the management of PIU states.

- ADCLK generator

  The ADCLK generator generates the clock for the A/D converter.

- AD interface controller

  The AD interface controller performs the control of the external circuit.

19.1.2 Scan Sequencer State Transition

Figure 19-4. Scan Sequencer State Transition Diagram

[Explanation of each state]

Disable state

The state where power to the external circuit can be turned off. The output pin in the High-z state and the input pin, in the masked state (the state where no misoperation will not occur if unstable input is received) so that power to the external circuit can be turned off.

Standby state

This is the state of waiting for scan. The external circuit is in the low-power-consumption state (no voltage is applied to the touch panel, the A/D converter is disabled). Usually, various modes are set in this state.

Caution Because the state shifts when the PIUSEQEN bit is made active, the PIUSEQEN bit must be activated only after the setting of various modes has completed.

MainBattCheck state

This is the state for measuring the voltage of the main battery. After obtaining voltage detection data by starting the A/D converter, DMA transfer is performed to the memory t generate DataRdyIntr. After the data transfer, the PIUSEQEN bit is activated automatically to go to the Standby state.

SubBattCheck state

This is the state for measuring the voltage of the sub battery. After obtaining voltage detection data by starting the A/D converter, DMA transfer is performed to the memory t generate DataRdyIntr. After the data transfer, the PIUSEQEN bit is activated automatically to go to the Standby state.

Command state

This is the state for operating the A/D converter by an arbitrary setting. Operation is the same as that of the Main/SubBattCheck state except that this state enables arbitrary setting of external circuit control signal PENCNT and command to the A/D converter, ADCMD. Setting of PENCNT and ADCMD is performed by the PIUCMDREG.

WaitPen Touch state

This is the state for waiting the "Touch" state of the touch panel. When the PIU has detected the "Touch" state, PenChgIntr, an internal interrupt of the PIU, occurs. In this case, if the PadAutoScan bit is active, the shift to the PenDataScan state occurs. If TClock stops during the WaitPen Touch state, the shift to the Suspend mode for detecting the state of the panel is enabled.

Note Conditions for the shift to the PenDataScan state

Because the occurrence of PenChgIntr and the detection of the condition for state shift have different timings, the, even if the "Touch" state is set when PenChgIntr occurs, the state shift does not occur in case where "Release" is set when the condition for state shift is detected. The timing of the detection of the condition for state shift is approx. 4 ADCLK after the occurrence of PenChgIntr.

PenDataScan state

This is the state for detecting coordinates on the touch panel. Four or five data for a one coordinate (when PADSCANTYPE = 1) are sampled by operating the A/D converter. DMA transfer to the memory occurs for each data and, after the data for one coordinate has been sampled, DataRdyIntr is generated.

Note 1. Because the scan sequencer does not stop even if the DMA request by PenIntr+PadStopAtPage or PenEndIntr is masked, overwrite of sampling data may occur on the PUDDATAREG. When an overwrite has occurred, PadDataLosIntr is generated.

2. When PadDataLosIntr has been generated, DataRdyIntr is generated even if the number of DMA transfers is less than the specified number and state shift occurs. However, if the DMA mask is set, DataRdyIntr is not generated and state shift cannot occur.

IntervalNextScan state

This is the state of waiting for the sampling time for the next coordinate and the "Release" state of the touch panel. When the detection of the state of the touch panel is performed and after the time set on the PIUSIVLREG has elapsed, the shift to the PenDataScan state occurs. When the PIU has detected the "Release" state within the preset time, PenChgIntr, an internal interrupt of the PIU, occurs. In this case, if the PADATSTOP bit is active, the shift to the PenDataScan state occurs and if it is inactive, the shift to the PenDataScan state occurs after the preset time has elapsed.

19.2 REGISTER SET

The following table lists the PIU registers.

Table 19-1. PIU Registers

Address	R/W	Register symbols	Function
0x0B00 0120	R/W	PIUDATAREG	PIU Touch Panel Point Data register
0x0B00 0122	R/W	PIUCNTREG	PIU Control register
0x0B00 0124	R/W1C	PIUINTREG	PIU Interrupt Cause register
0x0B00 0126	R/W	PIUSIVLREG	PIU Data Sampling Interval register
0x0B00 0128	R/W	PIUSTBLREG	PIU AD Converter Start Delay register
0x0B00 012A	R/W	PIUCMDREG	PIU AD Command register
0x0B00 013C	R/W	PIUCIVLREG	PIU AD Check Interval register

The function of each of these registers is explained in detail below.

19.2.1 PIUDATAREG

Figure 19-5. PIUDATAREG (0x0B00 0120)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	PAD DATA[11]	PAD DATA[10]	PAD DATA[9]	PAD DATA[8]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	PAD DATA[7]	PAD DATA[6]	PAD DATA[5]	PAD DATA[4]	PAD DATA[3]	PAD DATA[2]	PAD DATA[1]	PAD DATA[0]

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..12]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[11..0]	PADDATA[11..0]	Sampling data of A/D converter

This register indicates the sampling data received from the A/D converter.

19.2.2 PIUCNTREG

Figure 19-6. PIUCNTREG (0x0B00 0122) (1/2)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	PAD STOP	PENST P	PENST C	PAD STATE[2]	PAD STATE[1]	PAD STATE[0]	PADAT STOP	PADAT START
R/W	R/W	R/W	R	R	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	PADSC A NSTOP	PADSC A NSTAR T	PADSC A NTYPE	PIU MODE[1]	PIU MODE[0]	PIUSEQ EN	PIUPW R	PADRS T
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	W1
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15]	PADSTOP	Disables DMA request by PADINTR 1: Enabled 0: Disabled
D[14]	PENSTP	Previous touch panel pressing status 1: Pressed 0: Released
D[13]	PENSTC	Current touch panel pressing status 1: Pressed 0: Released
D[12..10]	PADSTATE[2..0]	Status of scan sequencer (See Figure 19-4) 111: Command 110: IntervalNextScan 101: PenDataScan 100: WaitPenTouch 011: SubBatteryCheck 010: MainBatteryCheck 001: Standby 000: Disable
D[9]	PADATSTOP	Sets automatic stop of sequencer when touch panel is

		released. 1: Samples one set of coordinate data and automatically stops sequencer when touch panel is released. 0: Does not automatically stop sequencer when touch panel is released.
D[8]	PADATSTART	Sets automatic start of sequencer when touch panel is pressed. 1: Automatically starts sequencer when touch panel is touched. 0: Does not automatically start sequencer when touch panel is touched.

Figure 19-6. PIUCNTREG (0x0B00 0122) (2/2)

Bit position	Bit name	Function
D[7]	PADSCANSTOP	Sets forced stop of sequencer. 1: Forcibly stops sequencer after one set of coordinate data has been sampled. 0: Does not stop sequencer.
D[6]	PADSCANSTART	Sets start of sequencer. 1: Forcibly starts sequencer. 0: Does not start sequencer.
D[5]	PADSCANTYPE	Enables sampling of pen pressure data. 1: Enabled 0: Disabled
D[4..3]	PIUMODE[1..0]	Sets PIU mode. 11: Detects voltage of subbattery. 10: Detects voltage of main battery. 01: Operates A/D converter by any command. 00: Samples panel coordinate data.
D[2]	PIUSEQEN	Enables operation of scan sequencer. 1: Enabled 0: Disabled
D[1]	PIUPWR	Sets PIU power mode. 1: Makes the output of PIU active and sets standby status. 0: Makes the output of PIU Hi-Z and allows external circuit to turn off power.
D[0]	PADRST	PIU reset 1: Reset 0: Normal

This register is used to make PIU settings.

The PADRST bit is automatically cleared to 0 4TClock after being set to 1.

19.2.3 PIUINTREG

Figure 19-7. PIUINTREG (0x0B00 0124)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	Reserved	Reserved	Reserved	PADENDINTR	PADINTR	PADDLOSTINTR	PADDRDYINTR	PADCHG INTR
R/W	R/W	R/W	R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Initial value	0	0	0	0	0	0	0	0

Bit position	Bit name	Function
D[15..5]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[4]	PADENDINTR	PIU DMA transfer 2-page boundary interrupt 1: Occurred 0: Normal
D[3]	PADINTR	PIU DMA transfer 1-page boundary interrupt 1: Occurred 0: Normal
D[2]	PADDLOSTINTR	PIUDATAREG data overwrite 1: Valid data overwritten 0: Normal
D[1]	PADDRDYINTR	PIU DMA transfer end interrupt 1: DMA transfer completed 0: Not completed
D[0]	PADCHGINTR	Change of touch panel contact status 1: Changed 0: Not changed

This register indicates the occurrence of various interrupts in the PIU.

19.2.4 PIUSIVLREG

Figure 19-8. PIUSIVLREG (0x0B00 0126)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	SCANINT VAL[10]	SCANINT VAL[9]	SCANINT VAL[8]
R/W	R	R	R	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SCANINT VAL[7]	SCANINT VAL[6]	SCANINT VAL[5]	SCANINT VAL[4]	SCANINT VAL[3]	SCANINT VAL[2]	SCANINT VAL[1]	SCANINT VAL[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	0	1	0	0	1	1	1

Bit position	Bit name	Function
D[15..11]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[10..0]	SCANINTVAL [10..0]	Sets coordinate data of touch panel sampling interval. Interval = SCANINTVAL[10..0] * 30 μs

This register sets the sampling interval for coordinate data of touch panel.

The value set by SCANINTVAL[10..0], multiplied by 30 μs is the sampling interval for one set of coordinate data. Logically, therefore, the sampling interval can be set in units of 30 μs within a range of 0 μs to 60810 μs (about 60 ms). Actually, however, the sampling interval will be equal to the time required to transfer one set of coordinate data if a sampling interval shorter than the time required to transfer the coordinate data is set.

19.2.5 PIUSTBLREG

Figure 19-9. PIUSTBLREG (0x0B00 0128)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	MODEL
R/W	R	R	R	R	R	R	R	R/W
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SELADCLK[3]	SELADCLK[2]	SELADCLK[1]	SELADCLK[0]	STABLE[3]	STABLE[2]	STABLE[1]	STABLE[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	1	1	1	0	1	1	1

Bit position	Bit name	Function
D[15..9]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[8]	MODEL	Selects conversion accuracy of A/D converter 1: 10 bits (TI TLV1543C) 0: 12 bits (TI TLC2543C)
D[7..4]	SELADCLK[3..0]	Sets ADCLK frequency 1111: 16.58 MHz / (4*SELADCLK[3..0] + 2) = 0.267 MHz : 0010: 16.58 MHz / (4*SELADCLK[3..0] + 2) = 1.658 MHz 0001: RFU 0000: RFU
D[3..0]	STABLE[3..0]	Panel application voltage stabilization wait time Wait time = STABLE[3..0] * 30 µs

This register sets the stabilization wait time for the voltage applied to the touch panel, the conversion accuracy of the A/D converter, and the type of the A/D converter externally connected.

The TI TLV1543C or TLC2543C can be connected to the VR4101. The A/D converter to be used is specified with the MODEL bit.

The clock supplied to the A/D converter is specified by the ADCLK[3..0] bits. The wait time for the voltage applied to the touch panel is set with STABLE[3..0], in a range of 0 µs to 450 µs in units of 30 µs.

19.2.6 PIUCMDREG

Figure 19-10. PIUCMDREG (0x0B00 012A)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	STABLEON	PENCNT [4]
R/W	R	R	R	R	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0	1

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	PENCNT [3]	PENCNT [2]	PENCNT [1]	PENCNT [0]	ADCMD [3]	ADCMD [2]	ADCMD [1]	ADCMD [0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	0	1	1	1	1	1	0

Bit position	Bit name	Function
D[15..10]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[9]	STABLEON	Enables the setting of touch panel application voltage stabilization wait time (set on STABLE[3:0] of PIUSTBLREG). 1: Waits for panel voltage stabilization time. 0: Ignores panel voltage stabilization time (voltage stabilization time = 0).
D[8..4]	PENCNT[4..0]	Output data during command scan 11110: Pen touch detected 11011: Panel voltage not applied 11010: X+ pin = High, X- pin = Low, Y- pin = measures voltage 11001: Y+ pin = High, Y- pin = Low, X- pin = measures voltage 10011: X+ pin = Low, X- pin = High, Y- pin = measures voltage 01011: Y+ pin = Low, Y- pin = High, X- pin = measures voltage Others: Reserved for future use. Operation is not guaranteed if any setting other than those above is made.
D[3..0]	ADCMD[3..0]	Sets A/D converter command. 1111: Reserved for future use. Operation is not guaranteed if this value is set. 1110: Power-down mode 1101: Vref+ 1100: Vref-

		1011: (Vref+ - Vref-)/2 1010: Selects input port (AIN10) : 0000: Selects input port (AIN0)

This register controls the PENCONT pin in command scan mode (when the PIUMODE bits of PIUCNTREG are 01) and sets the command of the A/D converter.

The value set for this register is effective only in command scan mode. It has no effect in other modes.

Because the TLV1543C A/D converter is not provided with a power-down mode, the setting of power-down mode by ADCMD[3..0] is ignored when the MODEL bit of PIUSTBLREG is set to 1.

19.2.7 PIUCIVLREG

Figure 19-11. PIUCIVLREG (0x0B00 013C)

Position	D15	D14	D13	D12	D11	D10	D9	D8
Name	Reserved	Reserved	Reserved	Reserved	Reserved	CHECKINTVAL[10]	CHECKINTVAL[9]	CHECKINTVAL[8]
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Position	D7	D6	D5	D4	D3	D2	D1	D0
Name	CHECKINTVAL[7]	CHECKINTVAL[6]	CHECKINTVAL[5]	CHECKINTVAL[4]	CHECKINTVAL[3]	CHECKINTVAL[2]	CHECKINTVAL[1]	CHECKINTVAL[0]
R/W	R	R	R	R	R	R	R	R
Initial value	1	0	1	0	0	1	1	1

Bit position	Bit name	Function
D[15..11]	Reserved	Reserved for future use. Write 0 to this bit. 0 is returned when this bit is read.
D[10..0]	CHECKINTVAL [10..0]	Interval count value

This register reads the real-time value of the internal register that sets the value of PIUSIVLREG and counts down.

19.3 REGISTER SETTING FLOW

The PIU requires initial settings before operating the scan sequence. In the case of initialization by a reset, the sequence intervals, etc. are set to the minimum speed, so they must be set again.

The registers that require initial setting and the setting procedures are as described below.

Register name	Bit name	Set value
PIUDATAREG	SCANINTVAL10:0	An arbitrary value
PIUSTBLREG	STABLE3:0	An arbitrary value

Further, settings for clearing DMA mask and interrupt mask are required outside the PIU register.

Table 19-2. Initial Settings at Scan Sequencer Operation

Initial setting	Unit name	Register name	Bit name	Set value
Clearing DMA mask	DCU	DMASENREG	DMASEN	1
	DCU	DMAMASKREG	DMAMASKPIU	1
Clearing interrupt mask	ICU	SYSINTREG	PIUINTR	1
	ICU	MPIUINTREG	bit[4:0]	0x1F
Clearing clock mask	CMU	CMUCLKMSK	MSKPIU	1

(1) Register setting flow for main battery voltage detection

Disable state

1) PIUCNTREG PIUPOWER = 1

↓

Standby state

2) PIUCNTREG PIUMODE [1:0] = 10

3) PIUCNTREG PIUSEQEN = 1

↓

MainBattCheck state

(2) Register setting flow for sub battery voltage detection

Disable state

1) PIUCNTREG PIUPOWER = 1

↓

Standby state

2) PIUCNTREG PIUMODE [1:0] = 11

3) PIUCNTREG PIUSEQEN = 1

↓

SubBattCheck state

(3) Register setting flow for automatic coordinate detection

Disable state

1) PIUCNTREG PIUPOWER = 1

↓

Standby state

2) PIUCNTREG PIUMODE [1:0] = 11
PADSCANTYPE = 0 or 1
PADAUTOSCAN = 1
PADAUTOSTOP = 1

3) PIUCNTREG PIUSEQEN = 1

↓

WaitPenTouch state

(4) Register setting flow for manual coordinate detection

Disable state

1) PIUCNTREG PIUPOWER

↓

Standby state

2) PIUCNTREG PIUMODE [1:0] = 00
PADSCANTYPE = 0 or 1
PADSCANSTART = 1

3) PIUCNTREG PIUSEQEN = 1

↓

PenDataScan state

(5) Register setting flow for shifting to the Suspend mode

Disable state

1) PIUCNTREG PIUPOWER = 1

↓

Standby state

2) PIUCNTREG PIUMODE [1:0] = 00
PADSCANTYPE = 0 or 1
PADSCANSTART = 0 ; Shift to the PadDataScan state when "Touch" is detected is prohibited.

3) DMAMASKREG DMAMASKPIU = 0 ; Setting DMA mask

4) MPIUINTREG PADCHGINTR = 1 ; Clearing PENCHGINTR interrupt mask

5) PIUCNTREG PIUSEQEN = 1

↓

WaitPenTouch state

(6) Register setting flow for restoring the Suspend mode

WaitPenTouch state

1) DMAMASKREG DMAMASKPIU = 1 ; Clearing DMA mask

2) PIUCNTREG PADAUTOSCAN = 1

↓

PenDataScan state

(7) Register setting flow for A/D converter control in the case of arbitrary setting

Disable state

1) PIUCNTREG PIUPOWER = 1

↓

Standby state

2) PIUCNTREG PIUMODE [1:0] = 01

3) Setting CNTREG, PENCNT and ADCMD

4) PIUCNTREG PIUSEQEN = 1

↓

Command state

19.4 OUTPUT TO PENCONT PINS

PENCONT[4..0] pins output the current state of the scan sequencer. The output data can be set on bit[8..4] of the PIUCMDREG.

Table 19-3. Relationship between PENCONT, ADSOUT, and State

State	PadState	PENCONT[4..0]	ADSOUT
Power off	Disable	-	-
Standby with low power consumption	Standby	11011 (0x1B)	1110 (0xE)
Pen state detection	WaitPenTouch/Interval	11110 (0x1E)	0011 (0x3)
Main battery voltage detection	MainBattCheck	11011 (0x1B)	0000 (0x0)

Sub battery voltage detection	SubBattCheck	11011 (0x1B)	0010 (0x2)
Y+=H,Y-=L,X=samp	PadDataScan	11001 (0x19)	0001 (0x1)
Y+=L,Y-=H,X=samp	PadDataScan	01011 (0x0B)	0001 (0x1)
X+=H,X-=L,Y=samp	PadDataScan	11010 (0x1A)	0011 (0x3)
X+=L,X-=H,Y=samp	PadDataScan	10011 (0x13)	0011 (0x3)

19.4.1 Order of Coordinate Data

The PIU makes the A/D converter as the object operate by cycles consisting of the number of sampling data + 1. The first A/D conversion operation cycle effects the analog/ digital conversion of the first data, the next cycle effects the first data transfer and the analog/digital conversion of the second data, and the last A/D converter cycle effects data transfer alone. Further, when switching the port to which the signal to control the voltage applied to the panel, PENCONT[4:0] is applied, a no apply state, where no voltage is applied to the panel, is provided to prevent any electrical short-circuiting in the panel.

(1) In the case of 4 data

Order	DMA transfer data	PENCONT[4:0]	ADSOUT
		11001 (0x19)	0001 (0x1)
		11011 (0x1B)	1110 (0xE)
1	Y+=H, Y-=L, X=samp	01011 (0x0B)	0001 (0x1)
		11011 (0x1B)	1110 (0xE)
	Y+=L, Y-=H, X=samp	11010 (0x1A)	0001 (0x3)
		11011 (0x1B)	1110 (0xE)
	X+=H, X-=L, Y=samp	10011 (0x13)	0011 (0x3)
		11011 (0x1B)	1110 (0xE)
	X+=L, X-=H, Y=samp	11011 (0x1B)	1110 (0xE)

(2) In the case of 5 data

Order	DMA transfer data	PENCONT[4:0]	ADSOUT
		11011 (0x1B)	1110 (0xE)
		11110 (0x1E)	0011 (0x3)
		11011 (0x1B)	1110 (0xE)
	Detection of pen state	11001 (0x19)	0001 (0x1)
		11011 (0x1B)	1110 (0xE)
	Y+=H, Y-=L, X=samp	01011 (0x0B)	0001 (0x1)
		11011 (0x1B)	1110 (0xE)
	Y+=L, Y-=H, X=samp	11010 (0x1A)	0001 (0x3)
		11011 (0x1B)	1110 (0xE)
	X+=H, X-=L, Y=samp	10011 (0x13)	0011 (0x3)
		11011 (0x1B)	1110 (0xE)
	X+=L, X-=H, Y=samp	11011 (0x1B)	1110 (0xE)
		11011 (0x1B)	1110 (0xE)

19.5 PIU OPERATION TIMINGS

19.5.1 Explanation of signals in the timing chart

(1) Internal signals

tclk_touch	Internal reference clock, TClock
tclk_maskr	Clock mask signal synchronized with the clock for the A/D converter
cspiub	PIU chip select signal
pird_piwrb	Internal bus read/write strobe signal
piad[3:0]	Internal address bus
piwrdata[15..0]	Internal write data bus
piuout[15..0]	Internal read data bus
drqpad	DMA request
drakpad	DMA acknowledge
page	Page boundary signal
padstat[2..0]	Sequencer state
penchgintr	penchgintr interrupt factor
piudatardyintr	piudatardyintr interrupt factor
datalostintr	datalostintr interrupt factor
penintr	penintr interrupt factor
penendintr	penendintr interrupt factor

(2) External pins

ADCLK	Reference clock for the A/D converter
ADCS*	Chips select signal for the A/D converter
ADEOC	Signal to indicate the completion of A/D conversion by the A/D converter
ADIN	Serial bus to transfer the converted data of the A/D converter
ADSOUT	Serial bus to transfer the channel select and other data of the A/D converter
PENCONT[4..0]	Signal to control the voltage to be applied to the touch panel
PENCHGINTR*	Interrupt signal to be input when the panel in the touch state

19.5.2 Battery Voltage Detection

In battery voltage detection, the input channel of the A/D converter is switched to the port to which the battery is connected to obtain 1 data (10/12 bits) of digital data and it is DMA transferred.

The attached drawing shows the timing chart for main battery voltage detection.

Figure 19-12. PIU Battery Voltage Detection Timing

19.5.3 Coordinate Detection

In coordinate detection, 1 set of coordinate data consisting of 4 or 5 data is obtained. As the timing, a cycle that is similar to that for battery voltage detection is performed successively.

Figure 19-3 shows the timing chart for 5-data coordinate detection.

Figure 19-13. PIU Coordinate Data Detection Timing at 5-data Operation (1/2)

(a) Interrupt detection cycle, A/D input port selection/conversion cycle

Figure 19-13. PIU Coordinate Data Detection Timing at 5-data Operation (2/2)

(b) A/D input port selection/conversion cycle, converted data transfer cycle 1

(c) Converted data transfer cycle4, 5

19.5.4 Page Boundary Interrupt

During DMA operation, a 1-page boundary interrupt (PENINTR) or a 2-page boundary interrupt (PENENDINTR) occurs when transferred data exceeds page boundaries of the DMA buffer.

Figure 19-14 shows the timing chart where, during the detection of a 4 data coordinate, the generation of a DMA request is suppressed because the second DMA transfer was on the page boundary of the first page (PIUCNTREG STOPATPAGE = 1 condition is also required) and data has been lost.

Figure 19-14. PIU Page Boundary Interrupt Timing

19.5.5 Data Lost

In the battery voltage detection and coordinate detection cycles, there may such a case where the data received from the AD converter cannot be DMA transfer but overwrite and lost (data lost) for the reasons such as that the DMA request cannot be issued or the time waiting for the DMA acknowledge is too long.

Caution The PIU counts the number of the data that were DMA transferred and, upon completion of the specified times of data transfer, ends the DataScan state (PADSTATE[2:0]=5) by generating PIUDataRdyIntr. However, because valid data is not exist in the case of data lost, the specified number of data is not reached and the DataScan state does not end. To prevent this, in the case where DMA transfer is made after the completion of the data transfer from the A/D converter, the DataScan state must be forcibly ended by generating PIUDataRdyIntr regardless with the number of data transferred.

Figure 19-15. PIU Data Lost Timing

19.5.6 Other cautions

(1) Difference between the sequencer stop request and stop timing during coordinate detection

During coordinate detection, the sequencer does not stop immediately if it is attempted to stop the sequencer with PADSCANSTOP of CNTREG or disable of PIUSEQEN. Because coordinate data represents one coordinate with 4 or 5 data, the boundary between coordinate data in the DMA buffer deviates unless the specified number of data is DMA transferred. Therefore, the sequencer does not stop unless the specified number of data is accepted if a stop request is given in the middle of operation. Judgment as to whether the sequencer has stopped or not is made by the timing when PIUDataRdyIntr is generated.

(2) DMA mask setting timing

Do not perform DMA mask setting in the state that allows DMA transfer. Otherwise, DMA acknowledge will not be returned and thus the sequence operation will be disabled. Perform DMA mask setting in the Standby state.

[MEMO]

CHAPTER 25 VR4101 COPROCESSOR 0 HAZARDS

The VR4100 CPU core avoids contention of its internal resources by causing a pipeline interlock in such cases as when the contents of the destination register of an instruction are used as a source in the succeeding instruction. Therefore, instructions such as NOP must not be inserted between instructions.

However, interlocks do not occur on the operations related to the CP0 registers and the TLB. Therefore, contention of internal resources should be considered when composing a program which manipulates the CP0 registers or the TLB. The CP0 hazards define the number of NOP instructions which is required to avoid contention of internal resources, or the number of instructions unrelated to contention. This chapter describes the CP0 hazards of the VR4100 CPU core.

The CP0 hazards of the VR4100 CPU core are equally or less stringent than those of the R4000; Table 25-1 lists the Coprocessor 0 hazards of the VR4100 CPU core. Code which complies with these hazards will run without modification on the R4000.

The contents of the CP0 registers or the bits in the “Source” column of this table can be used as a source after they are fixed.

The contents of the CP0 registers or the bits in the “Destination” column of this table can be available as a destination after they are stored.

Based on this table, the number of NOP instructions required between instructions related to the TLB is computed by the following formula, and so is the number of instructions unrelated to contention:

(Destination Hazard number of A) - [(Source Hazard number of B) + 1]

As an example, to compute the number of instructions required between an MTC0 and a subsequent MFC0 instruction, this is:

(5) - (3 + 1) = 1 instructions

Table 25-1. VR4101 Coprocessor 0 Hazards

Instruction or Event	Source		Destination	
	CP0 Data Used, Stage Used	No. of cycles	CP0 Data Written, Stage Available	No. of cycles
MTC0			CPR [rd]	5
MFC0	CPR [rd]	3		
TLBR	Index, TLB	2	PageMask, EntryHi, EntryLo0, EntryLo1	5
TLBWI TLBWR	Index or Random, PageMask, EntryHi, EntryLo0, EntryLo1	2	TLB	5
TLBP	PageMask, EntryHi	2	Index	6
ERET	EPC or ErrorEPC, TLB	2	Status [EXL, ERL]	4
	Status	2		
CACHE Index Load Tag			TagLo, TagHi, PErr	5
CACHE Index Store Tag	TagLo, TagHi, PErr	3		
CACHE Hit ops.	cache line	3	cache line	5
Load/Store	EntryHi [ASID], Status [KSU, EXL, ERL, RE], Config [K0], TLB	3		
	Config [AD, EP]	3		
	WatchHi, WatchLo	3		
Load/Store exception			EPC, Status, Cause, BadVAddr, Context, XContext	5
Instruction fetch			EPC, Status	4
exception			Cause, BadVAddr, Context, XContext	5
Instruction fetch	EntryHi [ASID], Status [KSU, EXL, ERL, RE], Config [K0]	2		
	TLB	2		
Coproc. usable test	Status [CU, KSU, EXL, ERL]	2		
Interrupt signals sampled	Cause [IP], Status [IM, IE, EXL, ERL]	2		
TLB shutdown			Status [TS]	2 (Inst.), 4 (Data)

Cautions 1. If the setting of the K0 bit in the Config register is changed to uncached mode by MTC0, the accessed memory area is switched to the uncached one at the instruction fetch of the third instruction after MTC0.

2. A stall of several instructions occurs if a jump or branch instruction is executed immediately after the setting of the ITS bit in the Status register.

Remarks 1. The instruction following MTC0 must not be MFC0.

2. The five instructions following MTC0 to Status register that changes KSU and sets EXL and ERL may be executed in the new mode, and not kernel mode. This can be avoided by setting EXL first, leaving KSU set to kernel, and later changing KSU.

3. There must be two non-load, non-CACHE instructions between a store and a CACHE instruction directed to the same primary cache line as the store.

The status during execution of the following instruction for which CP0 hazards must be considered is described below.

(1) MTC0

Destination: The completion of writing to a destination register (CP0) of MTC0.

(2) MFC0

Source: The confirmation of a source register (CP0) of MFC0.

(3) TLBR

Source: The confirmation of the status of TLB and the Index register before the execution of TLBR.

Destination: The completion of writing to a destination register (CP0) of TLBR.

(4) TLBWI, TLBWR

Source: The confirmation of a source register of these instructions and registers used to specify a TLB entry.

Destination: The completion of writing to TLB by these instructions.

(5) TLBP

Source: The confirmation of the PageMask register and the EntryHi register before the execution of TLBP.

Destination: The completion of writing the result of execution of TLBP to the Index register.

(6) ERET

Source: The confirmation of registers containing information necessary for executing ERET.

Destination: The completion of the processor state transition by the execution of ERET.

(7) CACHE Index Load Tag

Destination: The completion of writing the results of execution of this instruction to the related registers.

(8) CACHE Index Store Tag

Source: The confirmation of registers containing information necessary for executing this instruction.

(9) Coprocessor Usable Test

Source: The confirmation of modes set by the bits of the CP0 registers in the “Source” column.

Examples 1. When accessing the CP0 registers in User mode after the content of the CU0 bit of the Status register is modified, or when executing an instruction such as TLB instructions, CACHE instructions, or branch instructions which use the resource of the CP0.

2. When accessing the CP0 registers in the operating mode set in the Status register after the KSU, EXL, and ERL bits of the Status register are modified.

(10) Instruction Fetch

Source: The confirmation of the operating mode and TLB necessary for instruction fetch.

Examples 1. When changing the operating mode from User to Kernel and fetching instructions after the KSU, EXL, and ERL bits of the Status register are modified.

2. When fetching instructions using the modified TLB entry after TLB modification.

(11) Instruction Fetch Exception

Destination: The completion of writing to registers containing information related to the exception when an exception occurs on instruction fetch.

(12) Interrupts

Source: The confirmation of registers judging the condition of occurrence of interrupt when an interrupt factor is detected.

(13) Loads/Sores

Source: The confirmation of the operating mode related to the address generation of Load/Store instructions, TLB entries, the cache mode set in the K0 bit of the Config register, and the registers setting the condition of occurrence of a Watch exception.

Example When Loads/Stores are executed in the kernel field after changing the mode from User to Kernel.

(14) Load/Store Exception

Destination: The completion of writing to registers containing information related to the exception when an exception occurs on load or store operation.

(15) TLB Shutdown

Destination: The completion of writing to the TS bit of the Status register when a TLB shutdown occurs.

Table 25-2 indicates examples of calculation.

Table 25-2. Calculation Example of CP0 Hazard and the Number of Instructions Inserted

Destination	Source	Contending internal resource	Number of instructions inserted	Formula
TLBWR/TLBWI	TLBP	TLB Entry	2	5 - (2 + 1)
TLBWR/TLBWI	Load or Store using newly modified TLB	TLB Entry	1	5 - (3 + 1)
TLBWR/TLBWI	Instruction fetch using newly modified TLB	TLB Entry	2	5 - (2 + 1)
MTC0, Status [CU]	Coprocessor instruction which requires the setting of CU	Status [CU]	2	5 - (2 + 1)
TLBR	MFC0 EntryHi	EntryHi	1	5 - (3 + 1)
MTC0 EntryLo0	TLBWR/TLBWI	EntryLo0	2	5 - (2 + 1)
TLBP	MFC0 Index	Index	2	6 - (3 + 1)
MTC0 EntryHi	TLBP	EntryHi	2	5 - (2 + 1)
MTC0 EPC	ERET	EPC	2	5 - (2 + 1)
MTC0 Status	ERET	Status	2	5 - (2 + 1)
MTC0 Status [IE] Note	Instruction which causes an interrupt	Status [IE]	2	5 - (2 + 1)

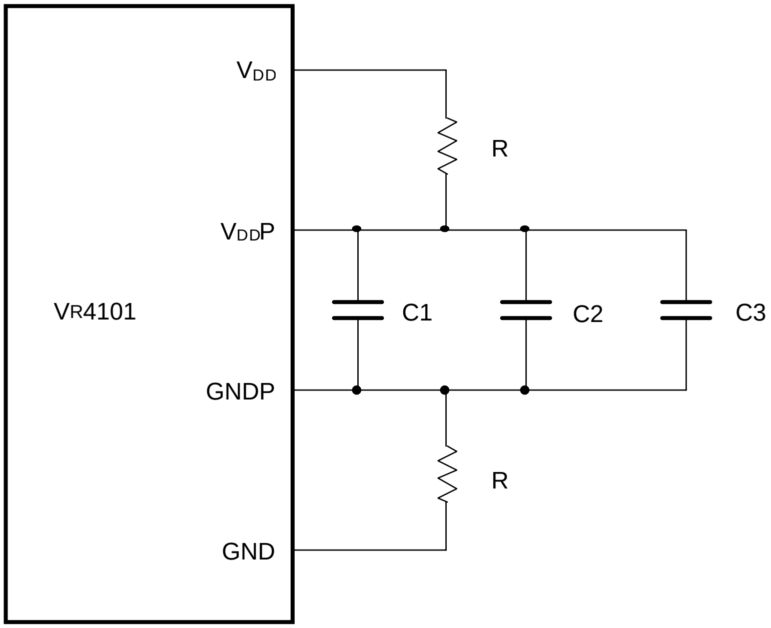
Note The number of hazards is undefined if the instruction execution sequence is changed by exceptions. In such a case, the minimum number of hazards until the IE bit value is confirmed may be the same as the maximum number of hazards until an interrupt request occurs which is pending and enabled.

[MEMO]

CHAPTER 26 PLL PASSIVE COMPONENTS

The Phase Locked Loop circuit requires several passive components for proper operation, which are connected to $V_{DD}P$ and GNDP as illustrated in Figure 26-1.

Figure 26-1. Example of Connection of PLL Passive Components



Remarks1. C1, C2, C3 capacitors and R resistors are mounted on the printed circuit board.

2. Since the value for the components depends upon the application system, the optimum values for each system should be decided after repeated experimentation.

It is essential to isolate the analog power and ground for the PLL circuit ($V_{DD}P$/GNDP) from the regular power and ground (V_{DD}/GND). Initial evaluations have yielded good results with the following values:

R = 5 Ω	C1 = 1 nF	C2 = 2 nF	C3 = 10 μF

Since the optimum values for the filter components depend upon the application and the system noise environment, these values should be considered as starting points for further experimentation within your specific application. In addition, the chokes (inductors: *L*) can be considered for use as an alternative to the resistors (*R*) for use in filtering the power supply.

[MEMO]

APPENDIX INDEX